Trabajo Fin de Grado
Grado en Ingeniería Informática

DESARROLLO Y DESPLIEGUE DE UNA INFRAESTRUCTURA DE EVALUACIÓN DE CALIDAD DEL SOFTWARE EN CLOUDS

AUTOR: IVÁN PLAZA ALONSO
TUTOR: FRANCISCO JAVIER GARCÍA BLAS
JESÚS CARRETERO PÉREZ

Leganés, Julio de 2015

Título: Desarrollo y despliegue de una infraestructura de evaluación de calidad del software en Clouds

Autor: Iván Plaza Alonso

Tutor: Francisco Javier García Blas
Jesús Carretero Pérez

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Trabajo de fin de grado el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

AGRADECIMIENTOS

Me gustaría agradecer a mis tutores Francisco Javier y Jesús Carretero todo el apoyo y ayuda que me ha ofrecido durante el desarrollo de este Trabajo.

A todos mis compañeros del laboratorio de ARCOS, con los que he pasado grandes momentos en este último año de Universidad.

A todos mis compañeros y profesores de la Asociación Musical San Roque, los que han conseguido convertir mi estrés y frustraciones en Arte.

Para mis padres, María Ángel y Juan Carlos, cuya educación y motivación me han permitido superar todas las metas que me he marcado en los últimos años.

A mis hermanos, Elisa e Ismael, cuyo cariño y apoyo a lo largo de todos estos años han hecho de mí la persona que soy hoy.

A Natalia, la persona que me ha acompañado más directamente en estos últimos años, su comprensión y cariño han logrado que hoy no vea un futuro sin ella.

¡¡MUCHAS GRACIAS!!

RESUMEN

Las empresas de desarrollo de software invierten muchos recursos en medir la calidad del código, con el fin de realizar productos software que se adapten a las necesidades reales de los clientes y que eviten fallos inesperados. En los últimos años se han desarrollado métodos de desarrollo ágil de software que potencian la colaboración interna de los desarrolladores, pero, ¿ofrecen métricas de medición de software?

En los últimos años se ha potenciado el uso de plataformas Cloud, debido a que ofrecen numerosos servicios de forma remota. Grandes empresas como IBM o Amazon venden esta tecnología a otras empresas, potenciando la comodidad, la capacidad de cómputo y la seguridad. Además, existen soluciones para crear *clouds* privados autogestionados. Una de las soluciones más conocidas actualmente es OpenStack, el cuál está compuesta de proyectos independientes con el objetivo de facilitar una instalación totalmente personalizada.

Además, existen muchas herramientas libres que facilitan a los usuarios el análisis de código y ofrecen los resultados de distintas formas: XML, texto plano, o de forma más gráfica en PDF o en una plataforma web. El único problema que ofrecen es que por lo general el análisis no está implementado de forma visual, por lo que exige en numerosas ocasiones introducir comandos muy largos o crear archivos con formatos específicos.

En este Trabajo Fin de Grado se detalla el proceso de creación de una infraestructura de evaluación de Software dentro de un sistema Cloud basado en OpenStack. El sistema desarrollado permite analizar el código fuente, en diversos lenguajes, y realizar un seguimiento continuo sobre la calidad del software.

Palabras clave: Cloud Computing, Calidad del Software, seguimiento.

ABSTRACT

Software development companies invest a lot of money to measure code quality, their principal purpose is to develop software products that adapt to clients real necessities and prevent unusual software errors. In the last years, agile software development methods have been created to improve developer's internal collaboration, but, does it offer software measures metrics?

Nowadays, cloud computing has been emerged like one of the most used systems because it offers a lot of remote services at a reduce cost. Big companies like IBM or Amazon offer its cloud computing technology to other companies enhancing comfort, computing power, and security. Also, there are solutions to create private clouds. One of the most popular cloud computing private systems is OpenStack. This solution offers a lot of independent projects with a customized installation.

Also, there are a lot of free tools that help users to analyse their code and shows the result in different formats like: XML, plain text, or more graphic in PDF format or in a web platform. But, it have an inconvenient, the inconvenient is that all the tools don't offers usually a visual or easy form to make the analysis, so many times it requires a length commands or create files with specific format.

This project details the process of creating an infrastructure to evaluated software. The infrastructure will be hosted on a cloud system, which is based on OpenStack. The system allows analysing source code in many languages and help developers to apply software quality metrics.

Key words: Cloud Computing, Software Quality, following.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

CAPITULO 1
INTRODUCCIÓN

En este capítulo se realiza la presentación del documento. Se describe de forma general la principal motivación que ha potenciado su creación, así como, los objetivos que pretende cubrir y la estructura que seguirá el trabajo.

1.1. MOTIVACIÓN

Actualmente, la mayoría de las empresas dedicadas al desarrollo de software invierten una considerable parte del presupuesto en analizar el software que generan con el objetivo de medir su calidad. Para realizar este análisis, se puede identificar dos formas de realizarlo: mediante pruebas estáticas y dinámicas (estos dos conceptos se definen en la Sección 2.2.). El principal objetivo de la medición de la calidad del software es el de generar aplicaciones cien por cien operativas y que no generen fallos inesperados durante su etapa de explotación.

La calidad del software se define [1],[2] como un conjunto de cualidades que caracterizan y determinan su utilidad, es decir, determinan su eficiencia, flexibilidad, portabilidad, seguridad, etc. La calidad se puede medir y variará dependiendo del sistema, programa o de la utilidad del mismo. Por este motivo, el software utilizado en aviación contará con un perfil de calidad que permita cero fallos, mientras que programas generados para un solo uso pasarán por un perfil menos restrictivo.

La medición del software puede ser realizada después de la generación del sistema o programa, pero puede generar gastos elevados al detectar errores que conlleven modificar el diseño inicial. Por este motivo, se recomienda realizar un seguimiento de la calidad en paralelo con la generación del software, ya que esto ayuda a evitar que los errores se propaguen y a tener un mayor control durante su ciclo de vida.

A lo largo de la historia, los gastos producidos por un fallo software superan muchas veces el precio de su desarrollo [3], como es el caso del error producido en el software del cohete Ariane 5 durante su vuelo inaugural en el año 1996. Tras 40 segundos de vuelo, el cohete sufrió una desviación de su ruta, lo que produjo que se partiera y explotara. El comité de investigación dictaminó que el fallo fue producido por un error en el software del sistema de referencia inercial. Este fallo consistía en que en un punto específico del código se realizaba una comparación errónea, un número en coma

flotante de 64 bits era tratado como entero de 16 bits. Además, las pruebas que se encargaban de detectar este tipo de errores nunca fueron pasadas.

Este tipo de errores no son cosa del pasado, ya que en el año 2010 la marca de automóviles Toyota se vio obligada a realizar una investigación sobre el sistema de frenado de uno de sus modelos. Tras la investigación, se dictaminó que más de 270.000 vehículos debían ser inspeccionados, ya que el software encargado de calibrar el sistema de frenado contenía un error.

Para evitar este tipo de errores, existen estándares de programación para realizar mediciones y corregir errores, como es el caso de las pautas MISRA [4] propuestas por la asociación de la confiabilidad del software de la industria del motor, o del conjunto de las normas ISO/IEC 25000 [5] que establecen un estándar internacional para realizar evaluaciones sobre la calidad y la evaluación de los productos software.

Además, existen herramientas como SonarQube que ofrece a los usuarios un entorno *web*, en el que poder visualizar los resultados del análisis de la calidad del código. Los usuarios pueden comprobar la evolución de la calidad a lo largo del ciclo de vida del software del producto. El único problema que ofrece esta herramienta es que el proceso de análisis no está automatizado, y no ofrece una interfaz para realizar el análisis de una forma cómoda y sencilla.

Por todo esto, la principal motivación de este Trabajo Final de Grado es la creación de una plataforma que permita realizar evaluaciones sobre el código y un seguimiento centralizado en el análisis de código mediante pruebas estáticas.

1.2. Marco regulador Técnico-Legal

En este apartado se desarrollan todas las medidas legales que se deben tener en cuenta a la hora de realizar el sistema que se propone en este TFG. Para realizar este desarrollo, se contará con los datos que se establecen el Instituto Nacional de Tecnologías de la Comunicación, S.A. (INTECO) en su 4º Monográfico[6].

En cuanto a los datos personales, se tratarán los aspectos legales relacionados con los datos que utilizará el sistema. Por ello es necesario hacer referencia al Artículo 8 de la LOPD (Ley Orgánica de Protección de Datos)[7]. Por este motivo se deberán tener en cuenta las siguientes consideraciones:

- No se crearán ficheros dentro del sistema que contengan datos de carácter personal como ideología, religión, creencias, origen racial o étnico, o vida sexual.
- En caso de el sistema almacene datos de carácter personal (ideología, religión, creencias, origen racial o étnico, o vida sexual), se pedirá al usuario que acepte una serie de términos que den derecho a la empresa que implemente el sistema para almacenar y acceder a dichos datos.

En cuanto a la utilización del software, es necesario analizar las características asociadas a la licencias, ya que el Artículo 10 de la LPI (Ley de Propiedad Intelectual) [8] establece que los programas de ordenador son objeto de propiedad intelectual. Además, hay que tener en cuenta que según establece el Artículo 11 de esta misma ley, los derechos de autor no se pierden al realizar cualquier tipo de transformación sobre una obra, por lo que al modificar un programa con derechos de autor no elimina los derechos que existen sobre dicho software. Por último, el Artículo 153 establece que en caso de incumplir cualquiera de los derechos de autor, o, actuar de prestador de servicios de intermediación, para realizar actos de piratería informática, conllevará una multa comprendida entre 150.000 y 600.000 euros.

1.3. Objetivos

Como ya se ha indicado anteriormente, el principal objetivo de este Trabajo Final de Grado es la creación de una infraestructura o sistema que permita evaluar la calidad del software proporcionado por los usuarios. Para conseguir este objetivo es necesario conseguir los siguientes objetivos secundarios:

- Gestión de usuarios y estructuración de tal forma que los datos de un usuario sean privados y no puedan ser consultados por ningún otro.
- Dotar a la infraestructura de un subsistema que se encargue de gestionar la subida de proyectos y las sucesivas versiones de proyectos.
- Facilitar a los usuarios el análisis del código subido en la plataforma.
- Facilitar a los usuarios el seguimiento de la calidad de los proyectos analizados.
- Desplegar la solución en una plataforma flexible.

1.4. Estructura

Este documento cuenta con un total de 10 capítulos. Estos capítulos siguen el orden básico establecido por las normas de redacción de los Trabajos de Final de Grado establecidas por la Universidad Carlos III de Madrid. Además, el texto del trabajo presenta la estructura general del proceso de desarrollo software. El fin de este orden es el de realizar un documento que facilite el seguimiento del desarrollo del sistema propuesto. Por todo esto, la estructura del documento es la siguiente:

- Capítulo 1 Introducción: en este capítulo se presenta las motivaciones que han iniciado el desarrollo de este Trabajo Final de Grado. Además, se exponen todos los objetivos a alcanzar con su realización.
- Capítulo 2 Estado del Arte: en este capítulo se exponen todas las herramientas utilizadas para el desarrollo del TFG. También se ofrecen las alternativas existentes en el mercado y las elecciones tomadas.

- Capítulo 3 Análisis del Sistema: para realizar el análisis del sistema, se describirá el desarrollo los casos de uso, los requisitos funcionales y no funcionales y la matriz de trazabilidad, que demuestra que todos los casos de uso están cubiertos por al menos un requisito.
- Capítulo 4 Diseño del Sistema: en este capítulo se realizará el diseño detallado del sistema. Primero, se mostrará la arquitectura que seguirá el entorno Cloud sobre el que se implementará el sistema. Después, se expondrá el diagrama de flujo de ejecución. A continuación, se presentarán los diagramas de componentes. Y por último, se realizarán los diagramas de secuencia derivados directamente de los diagramas de componentes.
- Capítulo 5 Implementación: en este capítulo se explicará todo el proceso seguido para implementar el sistema, desde el uso de la infraestructura virtual empleada hasta la codificación de cada componente.
- Capítulo 6: Ampliación de Reglas para SonarQube: se explicará la forma de ampliar las reglas que posee SonarQube mediante la utilización de un árbol de sintáxis abstracto y el lenguaje de representación Xpath.
- Capítulo 7 Pruebas: se especificará el plan de pruebas por el que pasará el sistema. Además se mostrará la matriz de trazabilidad para comprobar que todos los requisitos quedan cubiertos por al menos una prueba.
- Capítulo 8 Evaluación de la herramienta: se evaluará la herramienta mediante el análisis de tres laboratorios pertenecientes a asignaturas del grupo ARCOS, con el fin de obtener resultados que indiquen el funcionamiento y el rendimiento del sistema.
- Capítulo 9 Conclusión y líneas futuras: en este capítulo se ofrecen todas las conclusiones extraídas durante la realización del TFG. Además, se compararan con los objetivos marcados al comienzo y se expondrán diversas líneas futuras que asegurarán la permanencia del sistema en el tiempo.
- Capítulo 10 Planificación y Presupuesto: se mostrará la planificación marcada inicialmente y se comparará con los tiempos reales dedicados a cada tarea. Por último, se mostrará el presupuesto real del TFG.
- Capítulo 11 Bibliografía: se expondrán todas las referencias utilizadas para la creación de este proyecto.

CAPITULO 2
ESTADO DEL ARTE

En este capítulo se tratarán todas las tecnologías y aspectos más relevantes que se han tenido en cuenta para la realización de este Trabajo Final de Grado.

2.1. VIRTUALIZACIÓN

La virtualización es una tecnología con la cual se crean versiones virtuales de recursos tecnológicos como: un computador, un volumen de almacenamiento, etc., mediante el software. Para poder realizar esta tecnología se necesita de una computadora, la cual ofrece sus recursos para poder realizar la virtualización. A esta máquina se le denomina hipervisor. Cuando el recurso virtualizado es un ordenador por completo (microprocesador, memoria, periféricos, sistema operativo, etc.), se le denomina máquina virtual. Estas máquinas virtuales pueden realizar las mismas funcionalidades que un computador convencional sin que la capacidad de cómputo se vea significativamente afectada.

Actualmente se ha extendido la virtualización de servidores, debido que mejora la disponibilidad de los recursos, el aumento de la utilización de la *CPU* y sobre todo, la disminución del tiempo de respuesta ante un error por el cuál se interrumpe el funcionamiento normal del servidor.

A continuación se presentan todas las alternativas analizadas para realizar la virtualización del sistema sobre el cual será diseñado y desplegado el sistema presentado en este proyecto.

2.1.1. OPENSTACK

OpenStack [9] es un sistema *cloud* desarrollado actualmente por un gran conglomerado de empresas para gestionar sistemas *clouds*. Es capaz de gestionar el cómputo, almacenamiento y la red interna del sistema desde un panel de control web, lo que proporciona una administración cómoda, sencilla y eficiente.

2.1.1.1. Historia

OpenStack nace en el año 2010 [10] con las empresas Rackspace y NASA, ambas empresas anunciaron que estaban trabajando juntas en un nuevo *software IaaS* (Infraestructura como servicio). Con esta unión se intentaban solucionar los fallos que se habían detectado en el *software* Eucalyptus, el cual también gestiona sistemas *clouds*, y ofrecer una alternativa real y viable a Amazon Web Services. Cabe destacar que se trata de un *software* libre y de código abierto, por lo que la comunidad que lo rodea se ha visto aumentada en poco tiempo.

OpenStack está basado en un conjunto de proyectos o componentes [11] que se relacionan entre sí, y que tienen como objetivo el control de las distintas partes que lo componen. Estas partes son: el procesamiento, almacenamiento, recursos de red, imágenes, etc. Con cada nueva versión aumenta el número de estos proyectos, tal y como se muestra en la Tabla 1.

Fecha	Nombre	Componentes	Estado
Octubre 2010	Austin	Nova, Swift	Obsoleta
Febrero 2011	Bexar	Nova, Swift, Glance	Obsoleta
Abril 2011	Cactus	Nova, Swift, Glance	Obsoleta
Septiembre 2011	Diablo	Nova, Swift, Glance	Soporte
Abril 2012	Essex	Nova, Swift, Glance, Horizon, Keystore	Soporte
Septiembre 2012	Folsom	Nova, Swift, Glance, Horizon, Keystore, Quantum, Cinder	Soporte
Abril 2013	Grizzly	Nova, Swift, Glance, Horizon, Keystore, Quantum, Cinder	Soporte
Octubre 2013	Havanna	Nova, Swift, Glance, Horizon, Keystore, Quantum, Cinder, Heat, Ceilometer	Soporte
Abril 2014	Icehouse	Nova, Swift, Glance, Horizon, Keystore, Neutron, Cinder, Heat, Ceilometer, Trove	Soporte
Octubre 2014	Juno	Nova, Swift, Glance, Horizon, Keystore, Neutron, Cinder, Heat, Ceilometer, Trove Sahara,	En desarrollo
Abril 2015	Kilo		En planificación

Tabla 1: Historial de versiones de OpenStack.

2.1.1.2. Proyectos que forman OpenStack

Como se ha indicado anteriormente, OpenStack está formado por un conjunto de componentes, también llamados proyectos, los cuales permiten gestionar todo el entorno que rodea al sistema *cloud*. La relación que existe entre ellos es la que se representa en la Ilustración 1 [12]. Dicha figura se corresponde a la versión de OpenStack Havanna, sobre esta versión se desarrollará el sistema.

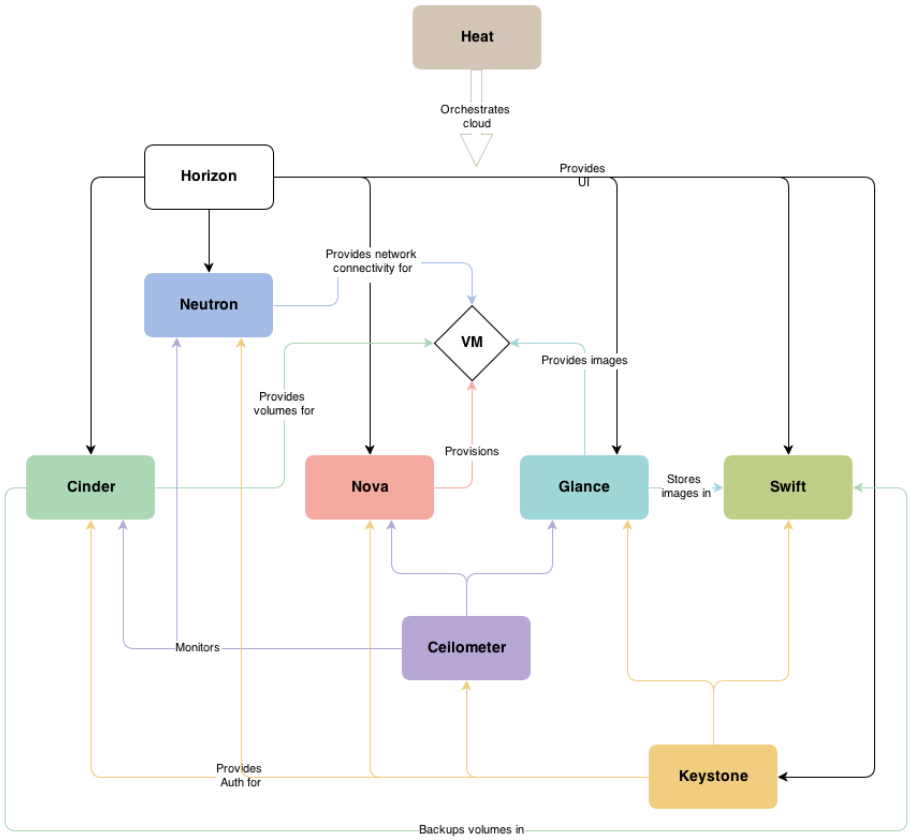


Ilustración 1: Arquitectura de OpenStack para la versión Havanna.

A continuación se pasa a describir cada uno de los componentes que forman esta versión de OpenStack de forma independiente.

Horizon: Representa la parte de administración de OpenStack, a este módulo también se le denomina *Dashboard*. Proporciona a los administradores y usuarios un entorno gráfico para gestionar los recursos del sistema *cloud*, tal y como se muestra en la Ilustración 2. Este módulo no es obligatorio en la instalación de OpenStack, ya que también se proporciona la *API* para que cada administrador del sistema se pueda construir su propia herramienta de gestión de comandos de consola.

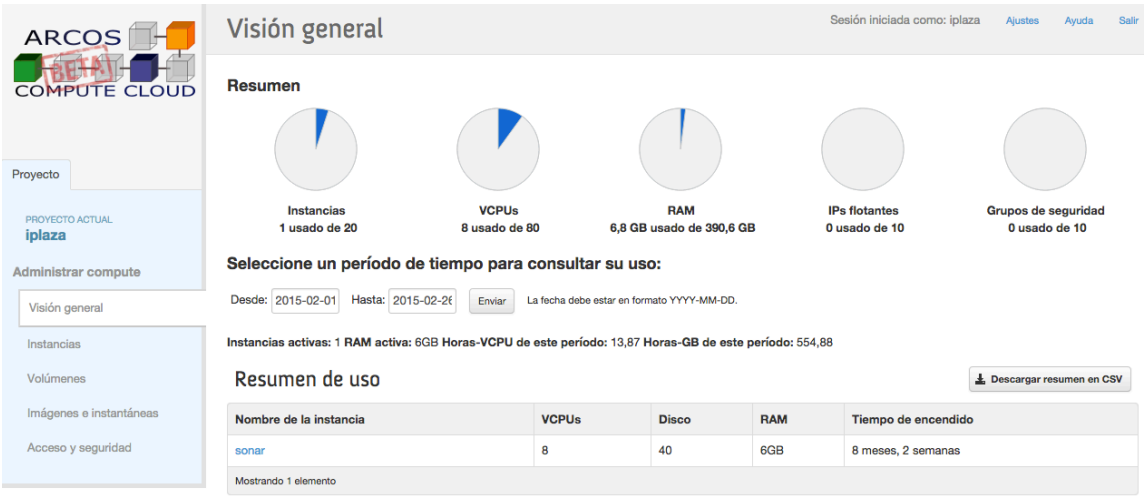


Ilustración 2: Panel de control del sistema OpenStack implementado en el grupo ARCOS.

Neutron: El proyecto Neutron, anteriormente llamado Quantum, es el encargado de gestionar las redes y las direcciones IP del sistema *cloud*. El sistema implementado en este proyecto hace que los administradores y los usuarios sean capaces de configurar su propia red, por lo que dota al sistema de un autoservicio real.

Un hecho muy destacable de este componente de OpenStack es que asegura que no se darán factores limitantes ni cuellos de botella. Algunas de las capacidades que presenta este componente son las siguientes:

- Aprovisionamiento de redes flexibles para adecuarlas a las exigencias de cada aplicación o grupos de usuarios.
- Inclusión de redes VLAN y redes planas para separar el tráfico de los servidores.
- Redirección dinámica de tráfico hacia cualquier recurso informático en caso de error.
- Permite la inclusión de servicios de red adicionales como sistemas de detección de intrusos (IDS), balanceo de carga, cortafuegos y redes virtuales. Todos estos servicios requieren una implementación y administración personalizadas.

Cinder: Este proyecto es el encargado de gestionar el almacenamiento a nivel de bloques de OpenStack. Este componente ha sido diseñado con el fin de permitir consumir los bloques de almacenamiento desde el proyecto de cómputo, Nova. Además, se proporciona una API y un panel dentro de la herramienta de gestión web para gestionar la creación, aplicación y desprendimiento de los dispositivos de bloque. Todo esto hace que los usuarios sean capaces de gestionar el almacenamiento que necesita cada una de sus instancias creadas en el *cloud*. Lo más destacable de este componente es que permite utilizar varias plataformas de almacenamiento además de la que presenta por defecto Linux, como son: Ceph, CloudByte, Coraid, EMC, etc.

Nova: Es el componente encargado de controlar la estructura del sistema de *cloud computing*, por lo que se trata del componente principal del sistema y de cualquier sistema *IaaS* (Infraestructura como servicio). Su diseño ha sido orientado para conseguir que se puedan gestionar los *pools* de recursos con los que cuentan cada equipo, así como para facilitar el uso de las tecnologías de virtualización como KVM, Xen, Hyper-V y LXC.

Glance: Este proyecto proporciona a los usuarios finales el control sobre el registro, localización y recuperación de las máquinas virtuales desplegadas en el sistema *cloud*. Además, permite almacenar un número ilimitado de copias de seguridad y de imágenes de discos. La API desarrollado proporciona una interfaz REST que permite realizar consultas sobre la información de cualquier imagen de disco y exportar las imágenes a otros servidores externos.

Ceilometer: Este componente es el encargado de proporcionar un sistema de facturación. Cuenta con una serie de contadores que se utilizan para mostrar el uso de recursos en el tiempo a los clientes. Este componente es opcional en la instalación de OpenStack.

Swift: Es el proyecto encargado del almacenamiento del sistema *cloud*. Todos los objetos y archivos del sistema son almacenados en los discos que conforman el *cloud* de forma redundante y escalable. Gracias a esto se consigue la integridad de los datos de todo el sistema. Además, previene la pérdida de información si se produce un fallo en cualquier disco duro.

Keystone: Representa el servicio de identidad del sistema, y es el encargado de dotarlo de un sistema de autenticación común para todo el *cloud*. Permite integrar servicios adicionales de directorios como LDAP. Además, cuenta con múltiples tipos de autenticación como: nombre de usuario y contraseña, *tokens* e inicios de sesión como en Amazon Web Service. Por último, provee al sistema de herramientas para restringir el acceso de los usuarios a los recursos.

2.1.2. ALTERNATIVAS

En este apartado se presentarán alternativas existentes en el mercado con las que el sistema de virtualización elegido compite actualmente.

2.1.2.1. Eucalyptus

Eucalyptus [13] es un *software* de código abierto que permite la creación de *clouds* privados compatibles con Amazon Web Service (AWS).

El código fuente está escrito en su mayoría en Java y está formado por más de 100 componentes. A su vez, proporciona una serie de servicios *web* que han sido modelados

para permitir la compatibilidad con AWS. Todo ello ha sido empaquetado en un sólo instalador que facilita la instalación y la utilización del sistema. Cabe destacar que Eucalyptus funciona bajo una infraestructura de virtualización formada por Linux y KVM.

2.1.2.2. VMware vCloud Suite

Se trata de un producto [17] desarrollado por la empresa VMware para la creación y gestión de clouds privados. Está basado en la plataforma de virtualización de servidores *vSphere*. Este producto está desarrollado fundamentalmente para la virtualización de servidores. Este producto potencia principalmente la creación de servidores virtualizados compatibles con un gran número de aplicaciones orientadas al mundo empresarial. Además, cuenta con integración para aplicaciones de Big Data. Al igual que OpenStack cuenta con un panel central o Dashboard, en el cual se puede realizar toda la gestión de utilización de recursos y ver el gasto producido.

2.1.2.3. IBM Bluemix

Es una plataforma *cloud* [18] desarrollada por IBM. Su principal propósito es ofrecer a los desarrolladores, tanto web como móvil, acceder a todo el software propio de IBM, así como software externo a IBM como MySQL, Eclipse, etc.

Bluemix está basado en la tecnología para sistemas *cloud* de código abierto *Cloud Foundry*. La idea principal que potencia IBM es la de dotar a los desarrolladores (tanto web como móviles) de facilidad y eficiencia al utilizar su Plataforma como Servicio (*PaaS*). Además, permite realizar despliegues personalizados por parte del desarrollador en la nube ocultando la mayor parte de las complejidades del proceso de instanciación.

2.1.3. RAZONAMIENTO DE LA DECISIÓN ADOPTADA

El principal motivo que motivó la utilización del sistema OpenStack fue que se trata de un sistema gratuito, muy acogido en diversas empresas y que cuenta con una gran comunidad de desarrolladores. El otro motivo es que el grupo de investigación de arquitectura de computadores, comunicaciones y sistemas (ARCOS) de la Universidad Carlos III de Madrid cuenta con una implementación totalmente operativa de este sistema *cloud*, el cual ha sido ofrecido para desplegar el sistema que se desarrolla en este proyecto final de carrera. La implantación del sistema Openstack también ha sido parte de desarrollo de este Trabajo Fin de Grado.

2.2. ANALIZADOR DE CÓDIGO

En este punto se presentan todas las alternativas analizadas para realizar el análisis del código. Es muy importante tener en cuenta que en el análisis de código se distinguen dos tipos:

- Análisis estático: este tipo de análisis se realiza sobre el código fuente, o el código objeto generado a partir de él, sin llegar a ejecutarlo.
- Análisis dinámico: este tipo de análisis se realiza sobre la ejecución del programa y observa el comportamiento que presenta a lo largo de la ejecución.

2.2.1. SONARQUBE

SonarQube [19] es una plataforma de código abierto que gestiona la calidad del código. La plataforma desarrollada cubre por completo los 7 ejes de la calidad del código:

- Arquitectura y diseño.
- Comentarios.
- Reglas de codificación.
- Errores potenciales.
- Complejidad.
- Pruebas unitarias.
- Código duplicado.

Además, cuenta con una interfaz web que permite observar de forma detallada el resultado del análisis del código. Sonar proporciona de una forma clara y concisa los errores potenciales que tiene el código, el porcentaje de los comentarios, cobertura con las pruebas unitarias, etc. En la Ilustración 3 se puede observar la forma en la que se le muestra a los usuarios la información relativa al análisis.

En cuanto al análisis, SonarQube permite analizar código escrito en distintos lenguajes de programación como: Java, C, C++, Cobol, PHP, JavaScript, etc., y realiza análisis dinámico y estático sobre el código deseado, atendiendo al perfil de reglas sobre las que se realiza el análisis.

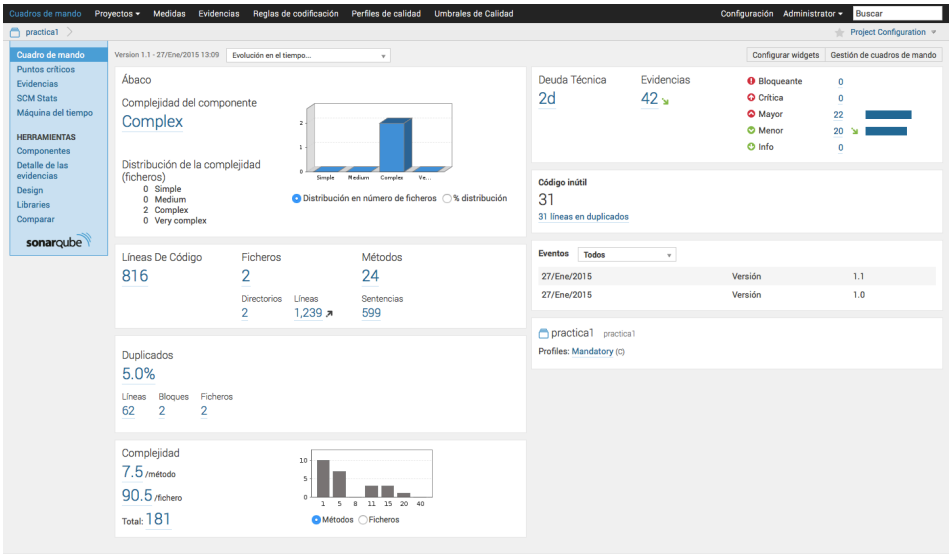


Ilustración 3: Cuadro de mando con los resultados del análisis en la versión 4.5.2 de SonarQube.

2.2.1.1. Arquitectura de alto nivel

La arquitectura que muestra Sonar es la representada en la Ilustración 4. En ella se puede ver que se compone de tres componentes:

- Base de Datos. Sonar necesita una base de datos donde almacenar todos los resultados de cada análisis. Las bases de datos soportadas por Sonar son:
  - Microsoft SQL Server 10.0 y 11.0.
  - MySQL 5.1 y 5.5.
  - Oracle 10G y 11G.
  - PostgreSQL 8.x y 9.x.
- Sonar-Runner. Esta es la aplicación proporcionada por Sonar para analizar el código y guardar los resultados en la base de datos.
- Interfaz Web. Es la interfaz *web* que proporciona Sonar para que los usuarios vean el resultado del análisis llevado a cabo.

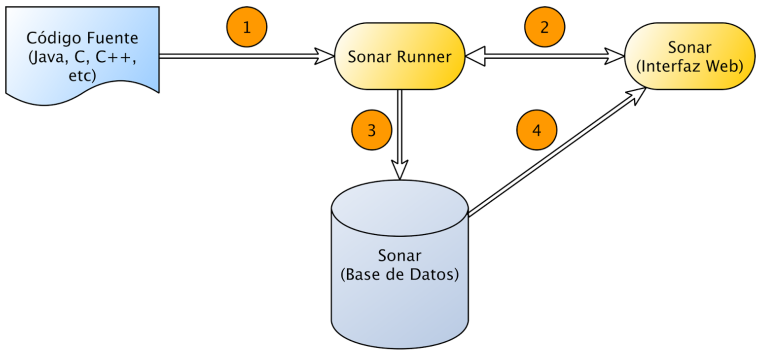


Ilustración 4: Arquitectura de alto nivel de Sonar.

2.2.1.2. Ampliación de las funcionalidades de Sonar

Sonar permite ampliar sus servicios mediante *plugins*. Estos *plugins* pueden ser descargados desde la interfaz web que proporciona Sonar tras su instalación. La inclusión de estos *plugins* permite ampliar métricas de calidad para el *software*, aumentar los lenguajes de programación soportados por la herramienta de análisis, cambiar el idioma del sistema, etc. Además, cada actualización que ofrece Sonar trae consigo nuevas funcionalidades, muchas de ellas dirigidas a dar una mayor cobertura al código.

2.2.2. ALTERNATIVAS

En este punto se pasan a analizar otras alternativas existentes para realizar análisis de código.

2.2.2.1. Cppcheck

Cppcheck [14] es un analizador de código estático de código libre orientado al análisis de código escrito en los lenguajes de programación C y C++. Este analizador tiene su base en detectar errores sintácticos en el código que muchos compiladores no detectan. Además, cuenta con una gran integración con las herramientas de desarrollo de software más utilizadas como: Eclipse, Visual Studio, Mercurial, etc.

Cppcheck se ejecuta mediante línea de comandos (en caso de no tenerlo integrado en un entorno de desarrollo) y su resultado se muestra como la salida del comando introducido, tal y como se muestra en la Ilustración 5. También permite guardar el resultado del análisis en un archivo XML para que pueda ser leído más fácilmente desde otras herramientas.

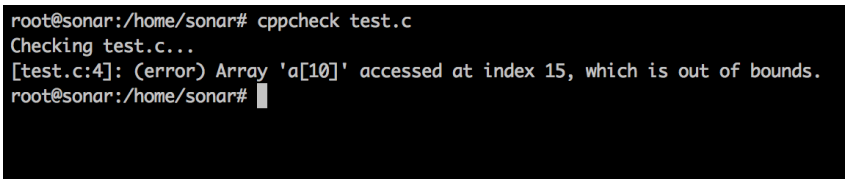


Ilustración 5: Ejemplo de salida de Cppcheck.

2.2.2.2. PMD

PMD [15] es un analizador de código abierto. Su misión es buscar fallos comunes como variables sin usar, creaciones innecesarias de objetos, etc. Soporta los siguientes lenguajes: Java, JavaScript, XML y XSL. Además, cuenta con una funcionalidad añadida que es la de detectar código duplicado, esta funcionalidad ha sido denominada CPD (*Copy-paste-detector*). Esta segunda herramienta tiene un mayor soporte de

lenguajes, ya que soporta Java, C, C++, C#, PHP, Ruby, Fortran y JavaScript. Al igual que Cppcheck, cuenta con plugins para integrarse en herramientas de desarrollo de software como: Maven, Eclipse, NetBeans, JBuilder, etc.

Su ejecución también se realiza mediante línea de comandos (si no se encuentra integrado en ningún entorno de desarrollo). El resultado es mostrado de forma predeterminada como salida del comando, o, en diversos formatos como: CSV, HTML, XML, etc.

2.2.2.3. CodePro Analytix

CodePro Analytix [16] es una herramienta desarrollada por Google para analizar código Java dentro del entorno de desarrollo Eclipse. Proporciona análisis de calidad del *software*, por lo que resulta especialmente útil para bajar los costes en el desarrollo del software y la planificación. Las características que presenta esta herramienta son las siguientes:

- Análisis de código: esta propiedad se encarga de detectar, notificar y reparar los errores detectados mediante las reglas de estilo definidas en los estándares de calidad del *software*.
- Métricas: se trata de una herramienta que muestra gráficamente el porcentaje de cumplimiento del código analizado con las reglas.
- Generación de pruebas JUnit: se trata de una funcionalidad que se encarga de generar automáticamente pruebas unitarias para cada componente desarrollado.
- Cobertura de código: permite medir la cobertura de código que muestra cada componente de forma individual (siempre que cuente con un método *main*). Además, permite visualizar los resultados de una forma gráfica y en forma de evolución con las distintas versiones.
- Análisis de dependencias: se trata de una herramienta totalmente gráfica que muestra las dependencias existentes entre proyectos, paquetes y clases.
- Análisis de código duplicado: se encarga de examinar y buscar estructuras codificadas duplicadas o similares dentro del código.

2.2.3. Razonamiento de la decisión adoptada

El principal motivo que impulsó la utilización de la herramienta SonarQube fue que es gratuito y de código abierto. Además, permite realizar análisis estático y dinámico de código, ampliar las reglas sobre las que analizar el código, crear perfiles personalizados, gestión de usuarios e incluir herramientas externas como Cppcheck y Valgrind. Además, esta es la alternativa que más lenguajes de programación cubre y, al igual que el resto de alternativas, no ofrece ninguna interfaz para realizar el análisis de la calidad del código.

2.3. LENGUAJES DE PROGRAMACIÓN WEB

En este apartado se analizarán varios lenguajes de programación para desarrollar aplicaciones *web*, y seleccionar un lenguaje, o un conjunto de lenguajes, que permita generar el sistema que se presenta en este trabajo.

2.3.1. HTML

El lenguaje HTML o *Hyper Text Markup Language* [20], es el lenguaje utilizado como estándar para la creación y representación de páginas *web*. Como es un lenguaje de hipertexto, el código se escribe de forma estructurada mediante etiquetas con las que se marca el inicio y el final de cada elemento que compone el documento. Cada documento HTML se compone solamente de texto, aunque se pueden referenciar desde él elementos como imágenes, vídeos, etc., por lo que otorga apariencia de ser un documento multimedia.

Para que un documento HTML pueda ser visualizado, es necesario el uso de un navegador, el cual interpretará el código de los distintos documentos HTML que componen la página web y se lo mostrará a los usuarios.

Es muy importante destacar que un documento HTML presenta al usuario elementos estáticos, es decir, invariables en el tiempo, por lo que si se desea actualizar la información de la página es necesario reescribir o modificar los documentos HTML que componen la página web.

2.3.2. JAVASCRIPT

Es un lenguaje de programación utilizado para crear páginas web dinámicas [21], por lo que representa un gran complemento al lenguaje HTML, el cual es totalmente estático. JavaScript se utiliza principalmente para ofrecer a las páginas web efectos visuales, animaciones, activaciones de botones y sobre todo mensajes emergentes para informar o avisar a los usuarios.

Además, este lenguaje es interpretado, por lo que no requiere ser compilado con anterioridad y su prueba se realiza directamente sobre el navegador tras realizar la codificación. El código escrito en JavaScript se ejecuta en el navegador del ordenador cliente, por lo que los servidores tienen menos carga computacional que con otro tipo de lenguajes como PHP y JSP, los cuales requieren que las páginas compiladas residan en su memoria. Pero, al estar el código en el ordenador del cliente, este es totalmente visible, por lo que puede acarrear problemas de seguridad.

2.3.3. PHP

Es un lenguaje de programación abierto muy utilizado en el desarrollo de páginas web [25], ya que permite ser incrustado dentro del código HTML. El principal uso de PHP es el de dotar de dinamismo a las páginas escritas en HTML mediante la incrustación de código PHP con alguna funcionalidad.

Su principal diferencia con JavaScript, el cual también añade dinamismo al código HTML, es que PHP se ejecuta en el lado del servidor, por lo que se genera un código HTML el cual se envía al cliente. Es muy importante destacar que el cliente recibe sólo código HTML, por lo que el código en PHP no será visible por el cliente.

Por último, cabe destacar que PHP se puede ejecutar en varios sistemas operativos, Linux, Windows, Mac OS, etc., pero, al estar orientado principalmente hacia entornos UNIX, es en este tipo de entorno donde se aprovechan mejor las funciones y su rendimiento.

2.3.4. JSP

Es una tecnología creada por Java orientada a crear páginas web dinámicas [23]. Permite crear páginas y aplicaciones web que pueden ser ejecutadas en varios servidores web. Además, como Java es multiplataforma, JSP también puede ser ejecutado en cualquier sistema operativo como Windows, Linux o Mac.

Al igual que en PHP, el código escrito en JSP se puede incrustar dentro de un documento HTML o XML mediante etiquetas que marcan el inicio/fin del fragmento de código. Su principal ventaja reside en el gran número de entornos de desarrollo que ofrecen herramientas de generación de proyectos web basados en JSP, lo que facilita las primeras fases de creación de proyectos *web*.

Esta tecnología precisa de un motor JSP, el cual se encarga de realizar todo el procesamiento interno de páginas JSP y de comprobar entre otras cosas que las páginas que piden los usuarios ya estén en memoria y se les sirva su código HTML equivalente.

2.3.5. RAZONAMIENTO DE LA DECISIÓN ADOPTADA

Para realizar la implementación del sistema propuesto se ha escogido una "mezcla" de los siguientes lenguajes de programación: HTML, JavaScript y PHP. Para defender esta decisión cabe destacar que en el mercado existen numerosas infraestructuras como LAMP (Linux, Apache, MySQL y PHP) que permiten realizar una instalación cómoda y eficiente de todas las herramientas que incorporan.

Además, el lenguaje de programación PHP es muy utilizado para el desarrollo de páginas web por su sencillez, su baja curva de aprendizaje y su documentación (con un gran número de ejemplos) publicada en internet.

2.4. SERVIDOR DE APLICACIONES WEB

Para poder interpretar el código escrito en PHP es necesario instalar un servidor web que interprete este lenguaje. De las opciones ofrecidas en el mercado destacan las que se exponen a continuación.

2.4.1. APACHE

Apache [24] es el servidor web de código abierto para el protocolo http más utilizado actualmente. Además de ser de código abierto, es multiplataforma, por lo que se puede instalar en varios sistemas operativos como: Windows, Linux Macintosh, etc.

El proyecto de Apache surge como una serie de mejoras hacia el servidor web NTCSA (ya en desuso). Las principales mejoras intentaban solucionar los errores que se iban conociendo sobre el servidor.

La principal ventaja con la que cuenta este servidor es que es de código abierto, lo que favorece el aumento de la comunidad de desarrolladores que lo rodea. Este hecho hace que todos los fallos que se detectan en la seguridad son corregidos muy rápidamente. Además, cuenta con un diseño modular, lo que favorece la sustitución o mejora de cada componente. Entre sus módulos se pueden destacar los siguientes:

- mod_ssl: módulo encargado de las conexiones seguras mediante el protocolo criptográfico TLS.
- mod_auth_ldap: módulo encargado de la autenticación de usuarios de un servidor LDAP.
- mod_security: módulo encargado de realizar un filtrado en el nivel de aplicación, de forma que aporta seguridad al servidor.
- mod_php: módulo encargado de compilar y ejecutar páginas dinámicas en PHP.

2.4.2. NGINX

Es un servidor HTTP de código abierto, libre, de alto rendimiento y muy ligero. Tiene la peculiaridad de que también actúa como proxy para los protocolos de correo electrónico IMAP y POP3. El servidor puede ser instalado en cualquier plataforma (Linux, Mac OS, Windows, etc.).

El alto rendimiento de este servidor viene determinado porque internamente cuenta con un solo proceso maestro y varios procesos de trabajo. El proceso maestro tiene la funcionalidad de leer la configuración del servidor y mantener los procesos de trabajo. Los procesos de trabajo son los encargados de realizar el procesamiento de las solicitudes que llegan al servidor.

En cuanto a las funcionalidades que ofrece este servidor se pueden destacar las siguientes:

- Caché para páginas web: el servidor cuenta con una caché propia para almacenar contenido estático y dinámico.
- Balanceo de carga: el servidor automatiza la distribución de las peticiones entre los distintos procesos de trabajo. Este balanceo es especialmente útil en servidores que cuentan con varios computadores distribuidos.
- Control del tiempo de ejecución: permite realizar configuraciones en las que se pueden reducir los tiempos de inactividad.
- Control de registro y errores: cuenta con un panel de control central que ofrece un entorno visual en el que se puede efectuar modificaciones sobre el acceso de usuarios, monitorización e incluso seguimiento de errores en las peticiones. En la Ilustración 6 [26] se puede observar una parte del monitor ofrecido por NGINX.

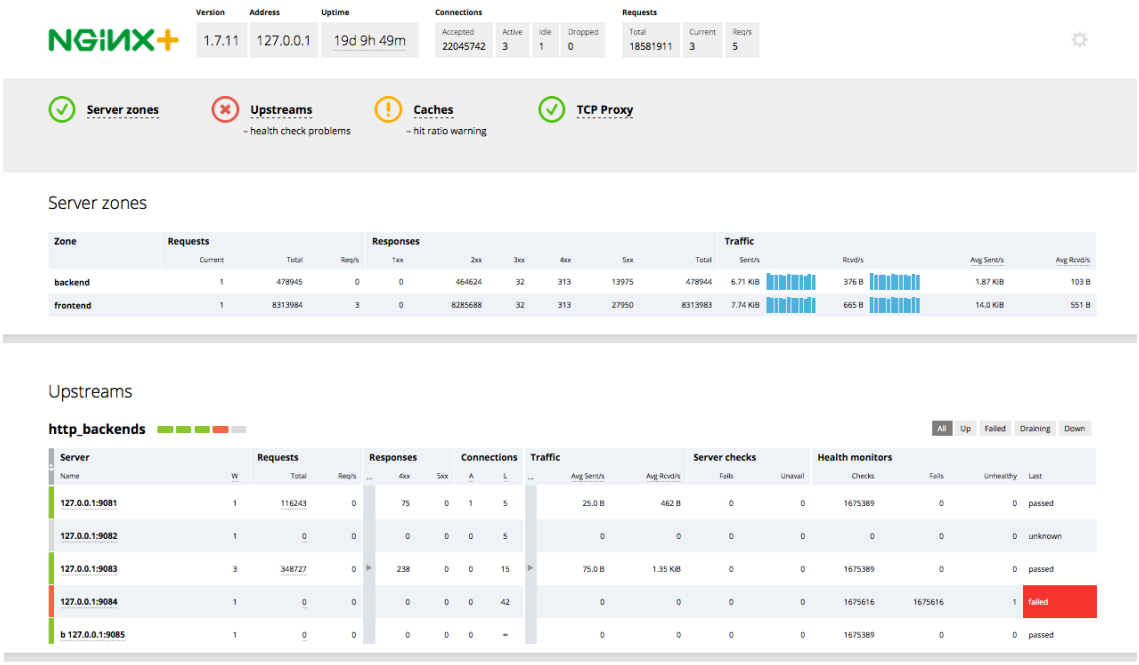


Ilustración 6: Panel de control del servidor NGINX.

2.5. SHELL SCRIPT

Los programas de *shell* [27] están formados por mandatos UNIX y sentencias de control de flujo, de asignación, funciones, etc. Cuentan con la peculiaridad de que este tipo de programas no son compilados, por lo que la *shell* del sistema UNIX se encarga de interpretarlos línea a línea.

Para que un archivo de texto sea considerado un script, en un sistema operativo tipo UNIX, es necesario que éste empiece con un encabezamiento específico. En caso de no incorporar el encabezamiento, se le puede indicar al sistema operativo que un fichero es un script mediante la extensión `.sh`.

Como se ha indicado anteriormente, la sintaxis utilizada en la codificación de este tipo de programas es la que se utiliza en la línea de comandos, por lo que un archivo con comandos de UNIX ejecutado como script ejecutará todos y cada uno de los comandos del archivo de forma ordenada de principio a fin.

2.6. GNUPLOT

Es un programa [31] para representar gráficas de funciones y superficies en diversos sistemas operativos, Linux, UNIX, Windows, Mac OS, etc., de código abierto. Cuenta con la peculiaridad de presentar los resultados directamente en la pantalla o almacenarlos en archivos gráficos como PNG, JPEG, SVG, etc. Su ejecución se puede realizar directamente sobre un terminal siguiendo la metodología presentada en los scripts. Cuenta con un lenguaje de programación propio, por lo que las estructuras de control como los bucles siguen una sintaxis determinada por los desarrolladores.

Además, cuenta con interfaces que permiten su implementación en otros lenguajes de programación como Python, Perl, Java, etc.

2.7. GOOGLE CHARTS

Se trata de una aplicación creada por Google que facilita la forma de presentar datos [32] en las páginas web, ya que proporciona una representación de datos sencilla al residir toda la lógica de procesamiento en los servidores de Google.

La gama que ofrece para representar gráficos es muy amplia y va desde los más básicos como: gráficos de barras y de líneas, hasta mapas geográficos más complejos.

La forma de utilizar este complemento en las páginas web es mediante una función JavaScript en la que se lee la biblioteca de Google Chart y se le indica que datos debe mostrar. Después, los datos son mostrados como una clase de JavaScript.

CAPITULO 3
ANÁLISIS DEL SISTEMA

En este apartado se realizará un análisis en profundidad de los componentes del sistema a desarrollar para conseguir así una especificación detallada del sistema. De esta forma, el sistema resultante de este apartado es el que pasará después por la fase de diseño. Primero, se desarrollará la idea sobre la que se centra el sistema, es decir, se explicará la funcionalidad básica del sistema sin llegar a utilizar una terminología muy específica y concreta. A continuación, se realizarán los casos de uso para conseguir identificar los procesos que realizarán los usuarios a los que va destinado el sistema. Después, se pasará a identificar todos los requisitos que debe cumplir el sistema, los cuales se extraen directamente de los casos de uso. Por último, se realizará una matriz para comprobar que los requisitos se ajustan y cubren todos los casos de uso presentados.

3.1. EXPOSICIÓN DEL SISTEMA

En este apartado se expondrá la idea que se quiere conseguir con el sistema. Como se ha indicado anteriormente, el sistema se deberá desplegar en un entorno de *cloud computing* y deberá ofrecer a los usuarios una interfaz clara y sencilla para realizar análisis de código, potenciando el seguimiento posterior de su calidad y su evolución a lo largo del desarrollo. Como el sistema será una plataforma web, y todos los datos de los usuarios serán almacenados en el servidor, se deberá dotar al sistema de las herramientas necesarias para la creación y modificación de cuentas de usuarios, por lo que los datos de un usuario serán privados. Además, los usuarios deberán subir su código a la plataforma para poder analizarlo posteriormente. Por este motivo, será necesario desarrollar una interfaz cómoda y sencilla que permita realizar subidas de código a los usuarios. Como mejora destacable a las herramientas existentes en el mercado, se deberá facilitar al usuario la forma de realizar los análisis, por lo que la iteración del usuario deberá ser mínima, y se le deberán mostrar de forma muy breve las distintas opciones que tiene para realizar el análisis de su código. Por último, los usuarios deberán ver todos los resultados extraídos de los distintos análisis realizados, así como la evolución de la calidad de su código y realizar comparativas entre los distintos códigos analizados, todos ellos pertenecientes al mismo usuario.

Por último, cabe destacar la terminología que se va a utilizar a partir de aquí en el resto del documento:

- Proyecto: conjunto de archivos codificados, bajo el mismo lenguaje, que unidos entre sí desempeñan una o varias funciones determinadas. Un proyecto puede estar formado por sólo un archivo de código.
- Versión: se trata de una asignación de un número decimal a los distintos proyectos que se suben al sistema, de esta forma se indica el grado de desarrollo que tienen. Además, la asignación de versiones permitirá realizar hacer un seguimiento continuo al desarrollo del sistema.
- Multi-proyecto: esta formado por varios proyectos, de esta forma al analizar un multi-proyecto se analizan varios proyectos en el sistema, con la peculiaridad de que además de obtener resultados individuales, por proyecto, se obtiene también un resultado general o resumen, que indica la calidad media de todos los proyectos analizados.
- Práctica: conjunto de archivos codificados por los alumnos que intentan resolver problemas expuestos por los profesores.
- Laboratorio: es el termino utilizado dentro del grupo de Arquitectura de Computadores, Comunicaciones y Sistemas (ARCOS), para denotar a las entregas que han realizado los alumnos a lo largo del curso. Si se atiende a su contenido, un laboratorio se compone de múltiples prácticas.
- Informe: Archivo en formato PDF que muestra al usuario la calidad de su software y del resto de métricas por las que ha pasado al ser analizado.
- Perfil de reglas: es el conjunto de reglas que selecciona el usuario para que su *software* sea analizado. Para un determinado lenguaje existirán varios perfiles que contengan más o menos reglas, las cuáles tendrán diferente nivel de restricción.

3.2. CASOS DE USO

Los casos de uso [33] sirven para obtener una descripción de los pasos que seguirán los usuarios para utilizar el sistema. Por este motivo, ofrecen una descripción a alto nivel de los procesos o actividades necesarias para desarrollar el sistema.

Los casos de uso serán desarrollados en diagramas siguiendo la notación que establece UML. A continuación se presentan los casos de uso con una breve descripción.

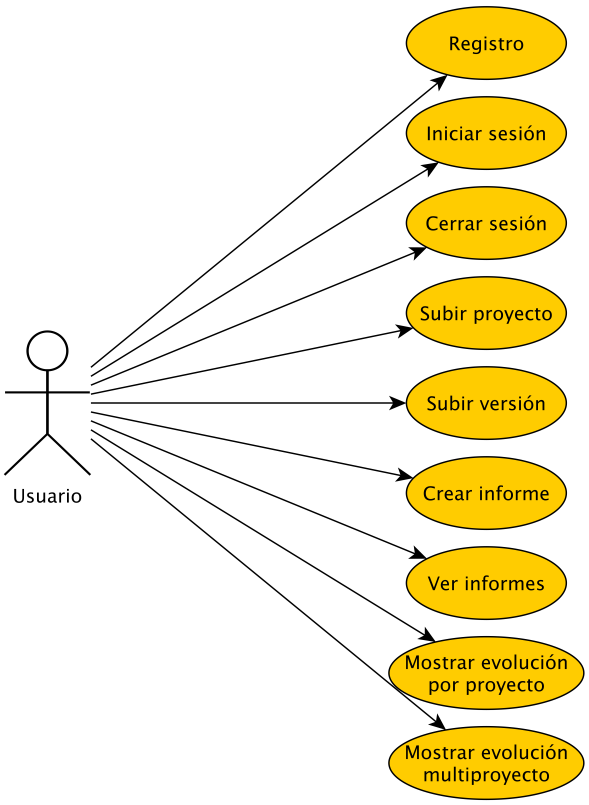


Ilustración 7: Casos de uso para un usuario conectado.

La Ilustración 7 muestra el primer contacto que tiene el usuario con el sistema, es decir, nada mas acceder al sistema tendrá que llevar a cabo el registro (si no se ha registrado antes) o el inicio de sesión. Al igual que acceso, al usuario se le debe brindar la forma de cerrar sesión dentro del sistema. Una vez dentro del sistema, el usuario podrá realizar las siguientes operaciones: subir proyectos, subir versiones de un proyecto, crear informe, ver los informes generados, mostrar la evolución de un proyecto y comparar varios proyectos.

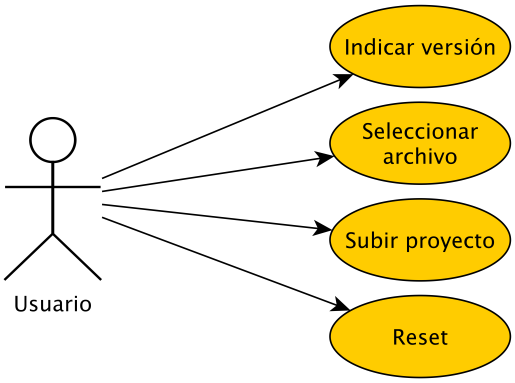


Ilustración 8: Casos de uso para realizar la subida de un proyecto.

En la Ilustración 8 se muestra el proceso que llevará a cabo un usuario para subir un proyecto al sistema, para ello, el usuario tendrá que indicar una versión y el archivo que

contiene el código fuente del proyecto. Además, se le brindará al usuario la opción de reiniciar el proceso en cualquier momento.

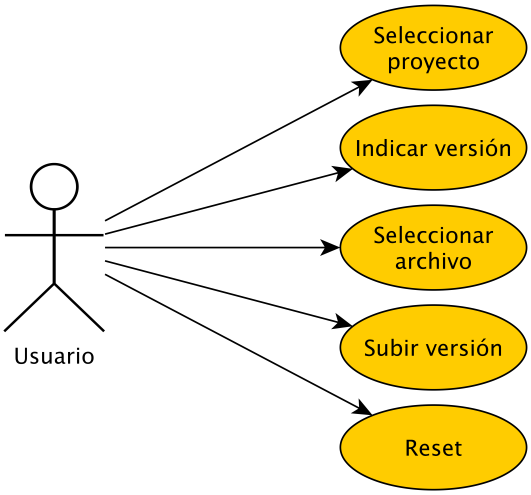


Ilustración 9: Casos de uso para realizar la subida de una versión.

Para que un usuario suba una nueva versión, cuenta con los casos de uso que se presentan en la Ilustración 9. Como se puede comprobar, el proceso de subida de una versión es muy parecido al de subida de proyectos, con la salvedad de que para subir una nueva versión es necesario seleccionar el proyecto.

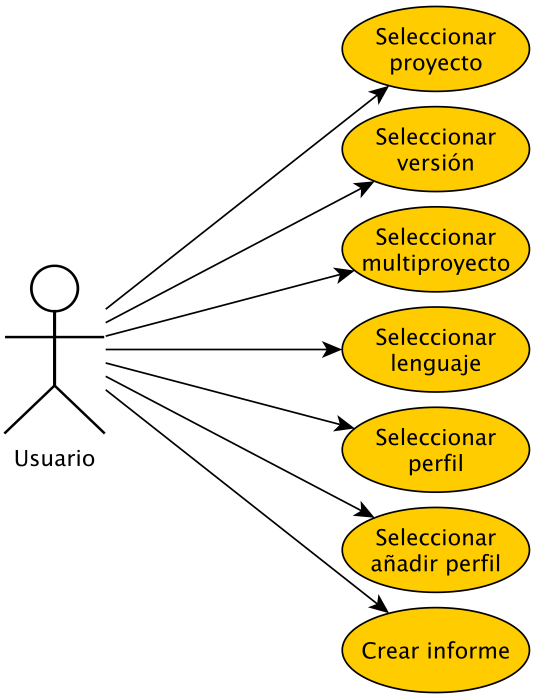


Ilustración 10: Casos de uso para generar un informe.

Para poder generar informes de proyectos subidos en el sistema, el usuario tendrá que realizar las operaciones que se muestran en la Ilustración 10. Tras proporcionar los

datos descritos en el caso de uso, el sistema internamente analizará el proyecto y generará un informe, el cual, será mostrado al usuario.

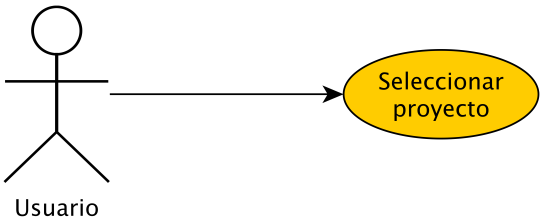


Ilustración 11: Casos de uso para visualizar la evolución de un proyecto.

Todo proyecto analizado queda almacenado en el sistema. Por este motivo, se le brinda al usuario la oportunidad de seguir de una manera más visual, que con los informes generados, la evolución del proyecto a lo largo de las versiones analizadas. En la Ilustración 11 se muestra el proceso que tiene que seguir el usuario para ver la evolución de un proyecto.

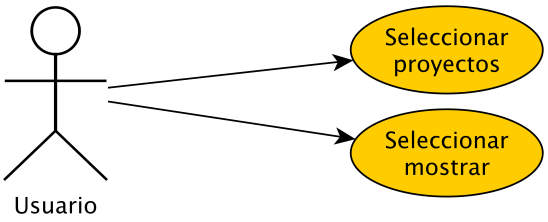


Ilustración 12: Casos de uso para comparar varios proyectos.

Además, se le proporciona al usuario la forma de que pueda comparar las métricas obtenidas para varios proyectos. La Ilustración 12 muestra los pasos que debe realizar el usuario para poder comparar varios proyectos.

Una vez vistos los distintos casos de uso se pasa a analizar cada uno de ellos de una forma más precisa haciendo uso de una serie de tablas. Todas las tablas siguen una estructura estándar, como se muestra en la Tabla 2.

ID	NOMBRE
Actores	
Descripción	
Precondiciones	
Poscondiciones	
Escenario básico	

Tabla 2: Modelo de tabla para los casos de uso.

Los campos necesarios para describir cada caso de uso son los siguientes:

- ID: cada caso de uso se identifica por un identificador y un nombre, en el caso del identificador seguirá la forma CU-XX, donde XX es un número que se irá incrementando en la unidad. El identificador deberá ser único ya que con él se pretende identificar a un solo caso de uso.
- Nombre: es el nombre que se le asocia al caso de uso.
- Actores: muestra a todas aquellas personas que realizarán la interacción mostrada por el caso de uso.
- Descripción: se trata de una descripción con la que se pretende mostrar la funcionalidad que se desarrolla.
- Precondiciones: son todas aquellas condiciones que se deben cumplir para poder realizar el caso de uso.
- Postcondiciones: son todos los efectos y/o consecuencias generados tras la ejecución del caso de uso.
- Escenario básico: en este apartado se muestra una descripción del flujo principal que se realiza cuando el actor interacciona con el sistema.

CU-01	REGISTRO DE USUARIOS
Actores	Usuario.
Descripción	El usuario se encuentra en la página de registro.
Precondiciones	El usuario se encuentra en la página de registro. El usuario introduce correctamente los datos que se le piden.
Poscondiciones	Se redirige al usuario al inicio del sistema.
Escenario básico	1. El usuario introduce la dirección del sistema. 2. El usuario selecciona registrarse en el sistema.

Tabla 3: Caso de uso 01.

CU-02	ACCESO AL SISTEMA
Actores	Usuario.
Descripción	El usuario se encuentra en la página de login.
Precondiciones	El usuario se encuentra en la página de login. El usuario introduce correctamente los datos que se le piden.
Poscondiciones	Se redirige al usuario al inicio del sistema.
Escenario básico	1. El usuario introduce la dirección del sistema. 2. El usuario introduce los datos correctamente. 3. El usuario selecciona entrar en el sistema.

Tabla 4: Caso de uso 02.

CU-03	FINALIZAR SESIÓN EN EL SISTEMA
Actores	Usuario.
Descripción	El usuario tiene una sesión iniciada en el sistema.
Precondiciones	El usuario se encuentra con la sesión iniciada en el sistema.
Poscondiciones	Se redirige al usuario a la página login.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona cerrar la sesión.

Tabla 5: Caso de uso 03.

CU-03	FINALIZAR SESIÓN EN EL SISTEMA
Actores	Usuario.
Descripción	El usuario tiene una sesión iniciada en el sistema.
Precondiciones	El usuario se encuentra con la sesión iniciada en el sistema.
Poscondiciones	Se redirige al usuario a la página login.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona cerrar la sesión.

Tabla 6: Caso de uso 04.

CU-05	SELECCIONAR SUBIR NUEVA VERSION
Actores	Usuario.
Descripción	El usuario selecciona la opción de subir una nueva versión de un proyecto al sistema.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se muestran las opciones para realizar la subida del proyecto.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción subir una nueva versión al sistema.

Tabla 7: Caso de uso 05.

CU-06	SELECCIONAR CREAR UN INFORME
Actores	Usuario.
Descripción	El usuario selecciona la opción de crear un informe.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se muestran las opciones disponibles para realizar el informe.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de crear un informe.

Tabla 8: Caso de uso 06.

CU-07	SELECCIONAR VER UN INFORME
Actores	Usuario.
Descripción	El usuario selecciona la opción de ver un informe.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se muestran todos los proyectos para los que se ha creado un informe.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de ver un informe.

Tabla 9: Caso de uso 07.

CU-08	SELECCIONAR MOSTRAR EVOLUCIÓN DE UN PROYECTO
Actores	Usuario.
Descripción	El usuario selecciona la opción de la evolución de un proyecto.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se muestran todos aquellos proyectos que ya han sido analizados.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de ver la evolución de un proyecto.

Tabla 10: Caso de uso 08.

CU-09	SELECCIONAR MOSTRAR EVOLUCIÓN DE VARIOS PROYECTOS
Actores	Usuario.
Descripción	El usuario selecciona la opción de la evolución de varios proyectos.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se muestran todos aquellos proyectos que ya han sido analizados.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de ver la evolución de varios proyectos.

Tabla 11: Caso de uso 09.

CU-10	SELECCIONAR VERSION PARA SUBIR UN PROYECTO
Actores	Usuario.
Descripción	El usuario indica la versión del proyecto que va a subir al sistema.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir un proyecto.
Postcondiciones	Se muestra correctamente la versión indicada.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir un proyecto. 3. El usuario indica la versión que desea subir.

Tabla 12: Caso de uso 10.

CU-11	SELECCIONAR ARCHIVO PARA SUBIR UN PROYECTO
Actores	Usuario.
Descripción	El usuario indica el archivo comprimido que contiene el proyecto que va a subir al sistema.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir un proyecto.
Postcondiciones	Se muestra correctamente el archivo en el sistema para realizar la subida.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir un proyecto. 3. El usuario indica el archivo que quiere subir al sistema.

Tabla 13: Caso de uso 11.

CU-12	REALIZAR SUBIDA DE UN PROYECTO AL SISTEMA
Actores	Usuario.
Descripción	El usuario, una vez ha indicado todos los campos indicados en el formulario, presiona sobre el botón de subir.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir un proyecto.
Postcondiciones	Se muestra la página con los valores por defecto y un mensaje indicando que la subida del archivo se ha producido de forma satisfactoria.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir un proyecto. 3. El usuario rellena el formulario de subida. 4. El usuario pulsa el botón subir.

Tabla 14: Caso de uso 12.

CU-13	RESETEAR EL PROCESO DE SUBIR UN PROYECTO
Actores	Usuario.
Descripción	El usuario presiona sobre el botón Reset para reiniciar el proceso de subida de un proyecto.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir un proyecto.
Postcondiciones	Todos los campos del formulario se reestablecen al valor por defecto.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir un proyecto. 3. El usuario pulsa el botón Reset del formulario de subida.

Tabla 15: Caso de uso 13.

CU-14	SELECCIONAR UN PROYECTO PARA SUBIR UNA NUEVA VERSION
Actores	Usuario.
Descripción	El usuario selecciona el proyecto del que quiere subir una nueva versión.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir una versión.
Postcondiciones	En el campo versión aparece como valor mínimo la versión más actual del proyecto más un incremento de 0.01.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir una versión. 3. El usuario selecciona un proyecto en el desplegable de proyectos.

Tabla 16: Caso de uso 14.

CU-15	SELECCIONAR VERSION PARA SUBIR UNA VERSION
Actores	Usuario.
Descripción	El usuario indica la versión del proyecto que va a subir al sistema.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir una nueva versión.
Postcondiciones	Se muestra correctamente la versión indicada.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir un proyecto. 3. El usuario selecciona un proyecto en el menú desplegable. 4. El usuario indica la versión que desea subir.

Tabla 17: Caso de uso 15.

CU-16	SELECCIONAR ARCHIVO PARA SUBIR LA NUEVA VERSION
Actores	Usuario.
Descripción	El usuario indica el archivo comprimido que contiene la nueva versión que se subirá al sistema.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir una versión.
Postcondiciones	Se muestran correctamente el archivo en el sistema para realizar la subida.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir una versión. 3. El usuario indica el archivo que quiere subir al sistema.

Tabla 18: Caso de uso 16.

CU-17	REALIZAR SUBIDA DE UNA VERSION AL SISTEMA
Actores	Usuario.
Descripción	El usuario, una vez ha indicado todos los campos indicados en el formulario, presiona el botón de subir.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir una versión.
Postcondiciones	Se muestra la página con los valores por defecto y un mensaje indicando que la subida del archivo se ha producido de forma satisfactoria.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir una versión. 3. El usuario rellena el formulario de subida. 4. El usuario pulsa el botón subir.

Tabla 19: Caso de uso 17.

CU-18	RESETEAR EL PROCESO DE SUBIR UNA VERSION
Actores	Usuario.
Descripción	El usuario presiona sobre el botón Reset para reiniciar el proceso de subida de un versión.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado subir una versión.
Postcondiciones	Todos los campos del formulario se reestablecen al valor por defecto.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de subir una versión. 3. El usuario pulsa el botón Reset del formulario de subida.

Tabla 20: Caso de uso 18.

CU-19	SELECCIONAR UN PROYECTO PARA CREAR UN INFORME
Actores	Usuario.
Descripción	El usuario selecciona el proyecto para crear un informe.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado crear informe.
Postcondiciones	En el desplegable de versiones se cargan todas las versiones subidas para el proyecto seleccionado.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de crear informe. 3. El usuario selecciona un proyecto en el desplegable de proyectos.

Tabla 21: Caso de uso 19.

CU-20	SELECCIONAR VERSION PARA CREAR EL INFORME
Actores	Usuario.
Descripción	El usuario indica la versión del proyecto para crear el informe.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado crear informe.
Postcondiciones	Se muestra correctamente la versión indicada.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de crear un informe. 3. El usuario selecciona un proyecto en el menú desplegable. 4. El usuario indica la versión que desea analizar.

Tabla 22: Caso de uso 20.

CU-21	SELECCIONAR MULTIPROYECTO PARA CREAR EL INFORME
Actores	Usuario.
Descripción	El usuario indica que el proyecto a analizar se compone de varios proyectos.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado crear informe.
Postcondiciones	Se queda activado el campo multiproyecto.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de crear un informe. 3. El usuario activa el campo multiproyecto.

Tabla 23: Caso de uso 21.

CU-22	SELECCIONAR EL LENGUAJE DEL PROYECTO PARA CREAR UN INFORME
Actores	Usuario.
Descripción	El usuario selecciona un lenguaje para crear un informe.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado crear informe.
Postcondiciones	En el desplegable de perfiles se cargan todos los perfiles de reglas asociados al lenguaje seleccionado.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de crear informe. 3. El usuario selecciona un lenguaje en el desplegable de lenguajes.

Tabla 24: Caso de uso 22.

CU-23	SELECCIONAR PERFIL DE REGLAS PARA CREAR UN INFORME
Actores	Usuario.
Descripción	El usuario indica el perfil de reglas sobre el que pasará el proyecto para crear el análisis.
Precondiciones	El usuario ha iniciado sesión en el sistema, ha seleccionado crear informe y un lenguaje de programación.
Postcondiciones	Se muestran los perfiles disponibles en el sistema asociados al lenguaje seleccionado.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de crear un informe. 3. El usuario selecciona un lenguaje en el menú desplegable. 4. El usuario indica el perfil por el que va a pasar el proyecto para crear el informe.

Tabla 25: Caso de uso 23.

CU-24	SELECCIONAR AÑADIR PERFIL DE REGLAS AL CREAR UN INFORME
Actores	Usuario.
Descripción	El usuario activa la opción de incluir el perfil de reglas.
Precondiciones	El usuario ha iniciado sesión en el sistema, ha seleccionado crear informe.
Postcondiciones	Se queda activado el campo añadir perfil de reglas.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de crear un informe. 3. El usuario activa el campo incluir perfil de reglas.

Tabla 26: Caso de uso 24.

CU-25	CREAR UN INFORME
Actores	Usuario.
Descripción	El usuario indica el proceso de creación de un informe.
Precondiciones	El usuario ha iniciado sesión en el sistema, ha seleccionado crear informe, ha rellenado todos los campos del formulario y selecciona crear el informe.
Postcondiciones	Se muestra al usuario un archivo pdf con el resultado del análisis del proyecto en un informe.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de crear un informe. 3. El usuario rellena el formulario de creación de un informe. 4. El usuario pulsa sobre el botón de crear un informe.

Tabla 27: Caso de uso 25.

CU-26	SELECCIONAR UN PROYECTO PARA MOSTRAR SU EVOLUCIÓN
Actores	Usuario.
Descripción	El usuario selecciona el proyecto para mostrar su evolución.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado mostrar evolución.
Postcondiciones	Se muestra la evolución del proyecto a lo largo de las versiones que han sido analizadas.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de mostrar evolución. 3. El usuario selecciona un proyecto de la lista mostrada.

Tabla 28: Caso de uso 26.

CU-27	SELECCIONAR PROYECTOS PARA MOSTRAR SU EVOLUCIÓN
Actores	Usuario.
Descripción	El usuario selecciona uno o varios proyectos para mostrar su evolución.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado mostrar evolución multiproyecto.
Postcondiciones	Los proyectos seleccionados se quedan marcados en la lista.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de mostrar evolución multiproyecto. 3. El usuario selecciona uno o varios proyectos de la lista mostrada.

Tabla 29: Caso de uso 27.

CU-28	SELECCIONAR MOSTRAR EVOLUCIÓN MULTIPROYECTO
Actores	Usuario.
Descripción	El usuario pulsa el botón mostrar en la ventana de evolución multiproyecto.
Precondiciones	El usuario ha iniciado sesión en el sistema y ha seleccionado mostrar evolución multiproyecto.
Postcondiciones	Aparece la evolución para los proyectos seleccionados en la lista.
Escenario básico	1. El usuario tiene la sesión iniciada. 2. El usuario selecciona la opción de mostrar evolución multiproyecto. 3. El usuario pulsa sobre el botón mostrar.

Tabla 30: Caso de uso 28.

3.3. Requisitos del sistema

En esta sección se enumerarán cada uno de los requisitos necesarios para la realización de este Trabajo Fin de Grado. Las tablas siguen una plantilla preestablecida, la cual se muestra en la Tabla 31.

ID					
Nombre					
Descripción					
Prioridad	ALTA		Verificabilidad	ALTA	
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 31: Modelo de tabla para los requisitos del sistema.

Los campos necesarios para describir cada requisito son los siguientes:

- ID: cada uno de los requisitos cuenta con un identificador único, el cuál facilita su seguimiento en las fases posteriores. Para cada tipo de requisito se utiliza la siguiente nomenclatura:
  - RF-XX: identifica a todos los requisitos funcionales del sistema, donde XX es un número que se irá incrementando en la unidad y su valor mínimo será 01.
  - RR-XX: identifica a todos los requisitos de restricción del sistema, donde XX es un número que se irá incrementando en la unidad y su valor mínimo será 01.
  - RI-XX: identifica a todos los requisitos de interfaz del sistema, donde XX es un número que se irá incrementando en la unidad y su valor mínimo será 01.
  - RS-XX: identifica a todos los requisitos de seguridad del sistema, donde XX es un número que se irá incrementando en la unidad y su valor mínimo será 01.
- Nombre: es el nombre que se le asocia a cada requisito.
- Descripción: se trata de una descripción detallada del requisito.
- Prioridad: cada requisito cuenta con una medida para indicar la urgencia en resolver el requisito.
- Verificabilidad: este campo indica si el requisito puede ser verificado fácilmente con una prueba.
- Fuente: es una referencia a la persona por la cual el requisito es obtenido.
- Necesidad: indica si el requisito es esencial en el sistema, en cuyo caso no será negociable. Si el requisito no es esencial, puede estar sujeto a negociaciones con el tutor.

3.3.1. Requisitos funcionales

Este tipo de requisitos indican la funcionalidad esencial del sistema, así como su propósito. Deben cubrir completamente los casos de uso indicados anteriormente.

RF-01					
Nombre	Registro de usuarios.				
Descripción	El sistema permitirá el registro de usuarios.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 32: Requisito funcional 01.

RF-02					
Nombre	Acceso al sistema.				
Descripción	El sistema permitirá el acceso a todos los usuarios que hayan sido registrados.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 33: Requisito funcional 02.

RF-03					
Nombre	Cierre de sesión en el sistema.				
Descripción	El sistema permitirá cerrar sesión a los usuarios.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 34: Requisito funcional 03.

RF-04					
Nombre	Subida de proyectos al sistema.				
Descripción	El sistema mostrará los campos necesarios para realizar la subida de proyectos a los usuarios registrados.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 35: Requisito funcional 04.

RF-05					
Nombre	Subida de versiones al sistema.				
Descripción	El sistema mostrará los campos necesarios para la subida de nuevas versiones para un proyecto.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 36: Requisito funcional 05.

RF-06					
Nombre	Creación de informes.				
Descripción	El sistema mostrará los campos necesarios para crear informes con el resultado del análisis de los proyectos.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 37: Requisito funcional 06.

RF-07					
Nombre	Visualización de informes.				
Descripción	El sistema mostrará una lista con los informes generados por el usuario.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 38: Requisito funcional 07.

RF-08					
Nombre	Visualización de la evolución de un proyecto.				
Descripción	El sistema mostrará una lista con los proyectos analizados por el usuario.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 39: Requisito funcional 08.

RF-09					
Nombre	Comparación de medidas de varios proyectos.				
Descripción	El sistema mostrará una lista con los proyectos analizados con el fin de comparar sus medidas.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 40: Requisito funcional 09.

RF-10					
Nombre	Selección de versión para un proyecto.				
Descripción	El sistema permitirá introducir la versión de un proyecto para subirlo al sistema.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 41: Requisito funcional 10.

RF-11					
Nombre	Selección de un archivo con el contenido del proyecto a subir.				
Descripción	El sistema permitirá seleccionar un archivo con el contenido del proyecto para realizar la subida al sistema.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 42: Requisito funcional 11.

RF-12					
Nombre	Subida de un proyecto al sistema.				
Descripción	El sistema permitirá la subida de proyectos a usuarios registrados.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 43: Requisito funcional 12.

RF-13					
Nombre	Mensaje de subida de un proyecto.				
Descripción	El sistema mostrará al usuario un mensaje con el resultado de la subida del proyecto.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 44: Requisito funcional 13.

RF-14					
Nombre	Reestablecer proceso de subida de un proyecto.				
Descripción	El sistema permitirá al usuario resetear el proceso de subida de un proyecto.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 45: Requisito funcional 14.

RF-15					
Nombre	Selección del proyecto del que se subirá la nueva versión.				
Descripción	El sistema permitirá seleccionar el proyecto del que se subirá una nueva versión.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 46: Requisito funcional 15.

RF-16					
Nombre	Selección del campo numérico de la nueva versión.				
Descripción	El sistema permitirá introducir el número de la versión que se subirá al sistema.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 47: Requisito funcional 16.

RF-17					
Nombre	Selección de un archivo con el contenido de la nueva versión.				
Descripción	El sistema permitirá seleccionar un archivo con el contenido de la nueva versión para realizar la subida al sistema.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 48: Requisito funcional 17.

RF-18					
Nombre	Subir versión al sistema.				
Descripción	El sistema permitirá subir nuevas versiones de los proyectos existentes en el sistema.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 49: Requisito funcional 18.

RF-19					
Nombre	Mensaje de subida de una versión.				
Descripción	El sistema mostrará al usuario un mensaje con el resultado de la subida de una versión.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 50: Requisito funcional 19.

RF-20					
Nombre	Reestablecer proceso de subida de una nueva versión.				
Descripción	El sistema permitirá al usuario resetear el proceso de subida de una versión.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 51: Requisito funcional 20.

RF-21					
Nombre	Selección del proyecto para realizar el informe.				
Descripción	El sistema permitirá seleccionar el proyecto del que se creará un informe.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 52: Requisito funcional 21.

RF-22					
Nombre	Selección de versión para crear un informe.				
Descripción	El sistema permitirá seleccionar la versión de un proyecto para crear el informe.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 53: Requisito funcional 22.

RF-23					
Nombre	Análisis de un proyecto formado por varios proyectos.				
Descripción	El sistema permitirá analizar multiples proyectos al mismo tiempo.				
Prioridad	ALTA		Verificabilidad	ALTA	X
	MEDIA	X		MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	
	EQUIPO			DESEABLE	X
				OPCIONAL	

Tabla 54: Requisito funcional 23.

RF-24					
Nombre	Selección del lenguaje de un proyecto.				
Descripción	La aplicación permitirá seleccionar el lenguaje del proyecto.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 55: Requisito funcional 24.

RF-25					
Nombre	Selección del perfil de reglas para crear el informe.				
Descripción	La aplicación permitirá seleccionar el perfil de reglas por el que pasará el proyecto para crear el informe.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 56: Requisito funcional 25.

RF-26					
Nombre	Incluir datos del perfil en el informe.				
Descripción	La aplicación permitirá incluir los datos de todas las reglas que forman el perfil en el informe generado.				
Prioridad	ALTA		Verificabilidad	ALTA	X
	MEDIA	X		MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	
	EQUIPO			DESEABLE	X
				OPCIONAL	

Tabla 57: Requisito funcional 26.

RF-27					
Nombre	Generar informes de proyectos.				
Descripción	La aplicación permitirá generar informes.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 58: Requisito funcional 27.

RF-28					
Nombre	Mostar el informe generado.				
Descripción	El sistema mostrará el informe generado al usuario tras su creación.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 59: Requisito funcional 28.

RF-29					
Nombre	Mostrar evolución de un proyecto.				
Descripción	La aplicación mostrará la evolución que sigue un proyecto.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 60: Requisito funcional 29.

RF-30					
Nombre	Seleccionar proyectos para comparar datos.				
Descripción	El sistema permitirá seleccionar varios proyectos para comparar los valores mas relevantes de los informes generados.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 61: Requisito funcional 30.

RF-31					
Nombre	Mostrar comparación de proyectos.				
Descripción	El sistema mostrará la comparación de los proyectos seleccionados.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 62: Requisito funcional 31.

3.3.2. Requisitos de restricción

Este tipo de requisitos establecen todos aquellos límites que el sistema no puede sobrepasar.

RR-01					
Nombre	Desplegar el sistema en un entorno cloud.				
Descripción	El sistema será desplegado en un entorno cloud.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 63: Requisito de restricción 01.

RR-02					
Nombre	El sistema se replicará automaticamente.				
Descripción	El sistema será almacenado en un sistema de archivos con replicación automática de datos.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 64: Requisito de restricción 02.

RR-03					
Nombre	Memoria RAM.				
Descripción	El sistema contará con 6 Gigabytes de memoria RAM.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	
	EQUIPO	X		DESEABLE	X
				OPCIONAL	

Tabla 65: Requisito de restricción 03.

RR-04					
Nombre	Almacenamiento del sistema.				
Descripción	El sistema contará con 40 Gigabytes de almacenamiento.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	
	EQUIPO	X		DESEABLE	X
				OPCIONAL	

Tabla 66: Requisito de restricción 04.

RR-05					
Nombre	Versión de Java.				
Descripción	El sistema tendrá la versión 1.7.0_75 de Java instalada [31].				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 67: Requisito de restricción 05.

RR-06					
Nombre	Base de datos.				
Descripción	El sistema tendrá instalado como sistema gestor de base de datos MySQL.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 68: Requisito de restricción 06.

RR-07					
Nombre	Versión de MySQL.				
Descripción	La versión de MySQL será la 5.5.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 69: Requisito de restricción 07.

RR-08					
Nombre	Mecanismo de almacenamiento de datos.				
Descripción	El mecanismo de almacenamiento que tendrá instalado MySQL será InnoDB.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 70: Requisito de restricción 08.

RR-09					
Nombre	Login de usuario único.				
Descripción	El login de un usuario será único.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 71: Requisito de restricción 09.

RR-10					
Nombre	Archivo que contiene el proyecto.				
Descripción	El archivo que contiene el proyecto para subir debe tener la extensión .zip.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 72: Requisito de restricción 10.

RR-11					
Nombre	Exploradores WEB soportados.				
Descripción	El sistema será soportado por todos los exploradores WEB.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	X
	EQUIPO			DESEABLE	
				OPCIONAL	

Tabla 73: Requisito de restricción 11.

RR-12					
Nombre	Gráficos de barras.				
Descripción	La evolución del número de errores de un proyecto se mostrará mediante un gráfico de barras.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	
	EQUIPO			DESEABLE	X
				OPCIONAL	

Tabla 74: Requisito de restricción 12.

RR-13					
Nombre	Evolución de la complejidad.				
Descripción	La evolución de la complejidad se mostrará en una tabla.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	
	EQUIPO			DESEABLE	X
				OPCIONAL	

Tabla 75: Requisito de restricción 13.

3.3.3. Requisitos de interfaz

Este tipo de requisitos describen como el sistema se comunica con el entorno, es decir, describen como el sistema se comunica tanto con los usuarios como con el resto de aplicaciones que conforman el sistema.

RI-01					
Nombre	Logotipo.				
Descripción	El logotipo del grupo de investigación ARCOS estará siempre visible.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	
	EQUIPO			DESEABLE	X
				OPCIONAL	

Tabla 77: Requisito de interfaz 01.

RI-02					
Nombre	Barra lateral.				
Descripción	El sistema contará con una barra lateral con todas las opciones que se les ofrece al usuario.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR	X	Necesidad	ESENCIAL	
	EQUIPO			DESEABLE	X
				OPCIONAL	

Tabla 78: Requisito de interfaz 02.

3.3.4. Requisitos de seguridad

Este tipo de requisitos establecen cómo debe hacer frente el sistema ante amenazas externas, además, de que seguridad se les ofrece a los usuarios frente a fallos del sistema.

RS-01					
Nombre	Conexiones con el sistema.				
Descripción	Todas las conexiones que se realicen con el sistema se harán mediante el protocolo HTTPS.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 79: Requisito de seguridad 01.

RS-02					
Nombre	Almacenamiento de contraseñas.				
Descripción	Todas las contraseñas se almacenarán cifradas.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 80: Requisito de seguridad 02.

RS-03					
Nombre	Atributos en la request.				
Descripción	Todo atributo que se envie por la request será cifrado.				
Prioridad	ALTA	X	Verificabilidad	ALTA	X
	MEDIA			MEDIA	
	BAJA			BAJA	
Fuente	TUTOR		Necesidad	ESENCIAL	X
	EQUIPO	X		DESEABLE	
				OPCIONAL	

Tabla 81: Requisito de seguridad 03.

3.4. Matriz de trazabilidad

En este apartado se muestra una tabla que representa la relación existente entre los casos de uso y los requisitos funcionales del sistema. Como se puede comprobar en la Tabla 82.1 y Tabla 82.2, todos los casos de uso se encuentran cubiertos por al menos un requisito.

RF/CU	CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13	CU-14
RF-01	X													
RF-02		X												
RF-03			X											
RF-04				X										
RF-05					X									
RF-06						X								
RF-07							X							
RF-08								X						
RF-09									X					
RF-10										X				
RF-11											X			
RF-12												X		
RF-13												X		
RF-14													X	
RF-15														X
RF-16														
RF-17														
RF-18														
RF-19														
RF-20														
RF-21														
RF-22														
RF-23														
RF-24														
RF-25														
RF-26														
RF-27														
RF-28														
RF-29														
RF-30														
RF-31														

Tabla 82.1: Matriz de trazabilidad de requisitos funcionales y casos de uso.

RF/CU	CU-15	CU-16	CU-17	CU-18	CU-19	CU-20	CU-21	CU-22	CU-23	CU-24	CU-25	CU-26	CU-27	CU-28
RF-01														
RF-02														
RF-03														
RF-04														
RF-05														
RF-06														
RF-07														
RF-08														
RF-09														
RF-10														
RF-11														
RF-12														
RF-13														
RF-14														
RF-15														
RF-16	X													
RF-17		X												
RF-18			X											
RF-19			X											
RF-20				X										
RF-21					X									
RF-22						X								
RF-23							X							
RF-24								X						
RF-25									X					
RF-26										X				
RF-27											X			
RF-28											X			
RF-29												X		
RF-30													X	
RF-31														X

Tabla 82.2: Matriz de trazabilidad de requisitos funcionales y casos de uso.

CAPITULO 4
DISEÑO DEL SISTEMA

En este capítulo se presenta el diseño detallado del sistema a desarrollar. Para realizar este diseño se presentará la arquitectura del sistema cloud en el que será alojada la aplicación web y todos sus componentes. Después, se presentará un diagrama de flujo, con este diagrama se mostrará una descripción visual de las actividades que formarán el sistema. Tras el diagrama de flujo, se mostrarán los diagramas de componentes agrupados por funcionalidades, con el fin de aportar a este documento una mayor legibilidad. Después, se mostrarán los diagramas de secuencias extraídos de los diagramas de componentes. Por último, se mostrará el modelo de datos utilizados para realizar la gestión del sistema.

4.1. DEFINICIÓN DE LA ARQUITECTURA DEL SISTEMA

La arquitectura que seguirá este sistema será la del patrón Modelo Vista Controlador (MVC) [29],[30]. Este patrón de diseño separa la lógica de negocio de la interfaz de usuario facilitando de esta forma la funcionalidad, mantenibilidad y escalabilidad de un sistema.

Este patrón divide la lógica que persigue cualquier aplicación o sistema en tres niveles diferenciados:

- Modelo: en esta capa se presenta la lógica de negocio, es decir, aquí es donde se realiza el acceso a cualquier dato, por lo que actúa de intermediaria con la base de datos.
- Vista: esta capa es la encargada de mostrar al usuario la información haciendo uso de modelos gráficos.
- Controlador: está situada entre la capa de vista y de modelo. Esta capa es la encargada de controlar las acciones que realiza el usuario de la siguiente forma: primero, las acciones del usuario son recogidas e interpretadas como solicitudes para el modelo, después, los resultados obtenidos del modelo se pasan a la vista para que esta, a su vez se lo muestre al usuario.

El esquema por el que se rige este patrón es el representado en la Ilustración 13.

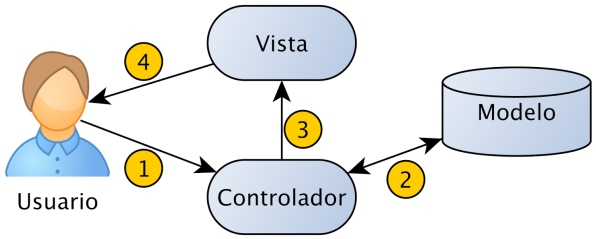


Ilustración 13: Esquema del patrón Modelo Vista Controlador (MVC).

Se pueden distinguir cuatro pasos en la realización de una consulta por parte del usuario:

1. El usuario realiza la petición al controlador. En el caso particular de este sistema, éstas se realizarán mediante el protocolo HTTP.
2. El controlador invoca al modelo para realizar dos operaciones posibles:
   - Actualización de los datos relativos al usuario.
   - Consulta de los datos del usuario.

   El modelo, cuando ha realizado la petición, retorna al controlador el resultado de la operación.
3. El controlador procesa los datos obtenidos del modelo y se los envía a la vista.
4. La vista presenta al usuario el resultado de la petición inicial que ha sido procesada por el sistema.

Las ventajas que ofrece este patrón son las siguientes:

- Permite una implementación modular, por lo que es posible añadir funcionalidades posteriormente mediante la creación de nuevos módulos.
- La vista siempre muestra al usuario información actualizada, ya que las vistas no contienen información, sino que muestran la información contenida en el modelo, el cual se actualiza automáticamente.
- A la hora de realizar modificaciones, como aumentar el número de métodos o realizar cambios en los datos autocontenidos, no implica realizar una modificación en el modelo y en la vista, dejando intacto por lo tanto el mecanismo de comunicación existente entre las capas.
- El hecho de modificar una vista sólo afecta a la forma de mostrar la información al usuario, dejando intacto el tratamiento que se realiza sobre los datos.

Por otra parte, este patrón también cuenta con desventajas, las cuales hay que tener en cuenta:

- El tiempo necesario para el desarrollo es más elevado que en otros patrones, ya que este patrón implica un mayor desarrollo de clases. Esta desventaja a la larga se convierte en una ventaja, ya que la mantenibilidad de este modelo se ve mejorada al contar con más clases.

- Necesita una arquitectura inicial sobre la que desarrollar tanto las clases como las interfaces.
- El modelo vista controlador está dirigido al diseño orientado a objetos. Este hecho hace que al utilizar lenguajes que no cuentan con este paradigma su desarrollo y posterior implementación sea mucho más costoso.

Los componentes software que consume el sistema se clasifican dentro del modelo MVC tal y como se muestra en la Ilustración 14.

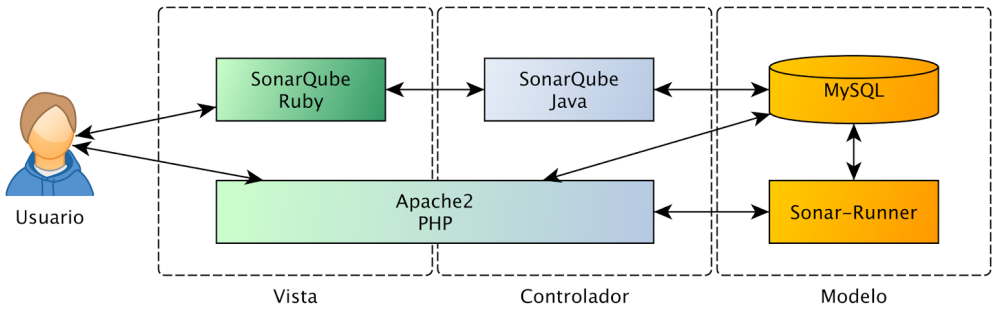


Ilustración 14: Clasificación de componentes dentro del modelo MVC.

Una vez comentado el patrón que seguirá el sistema, se pasa a realizar la arquitectura de alto nivel del sistema. Para ello, se presentará el entorno cloud en el que se desplegará el sistema.

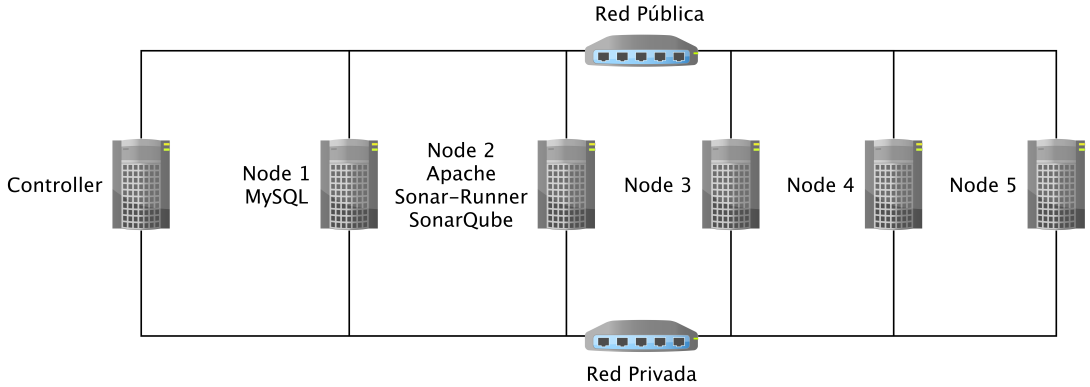


Ilustración 15: Arquitectura interna del sistema cloud.

El sistema será desplegado en un cloud que sigue la estructura que se muestra la Ilustración 15. En el cloud se pueden distinguir dos tipos de nodos:

- *Controller*: se trata del nodo que coordina todo el cloud. En él se realizan todas las operaciones de registro de nodos, gestión de direcciones IP, almacenamiento de imágenes de sistemas operativos, gestión de usuarios, creación de instancias, etc.
- *Node*: en este tipo de nodos se realiza la creación e instanciación de máquinas virtuales. Es en estos nodos en los que se realizará la instalación y despliegue de la infraestructura necesaria para el sistema. Como se puede comprobar en la

Ilustración 15, la base de datos estará alojada en una máquina dedicada. Con este hecho se pretende dotar al sistema de mayor seguridad intentando aislar los servicios de cada aplicación por separado.

4.2. DIAGRAMA DE FLUJO

En este apartado se desarrollará el diagrama de flujo [34] del sistema a implementar. En este diagrama se representa de una forma gráfica el funcionamiento y el flujo de información del sistema. Cabe destacar que en este punto no se desarrollará en profundidad el funcionamiento de cada una de las actividades, ya que ese nivel de detalle se alcanzará en las secciones de diagramas de componentes y de secuencia.

En cuanto a la notación utilizada para realizar el diagrama se ha seguido la que se muestra en la Tabla 83:

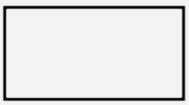
Símbolo	Significado
	Terminal de inicio: indica el inicio del flujo del sistema.
	Terminal de fin: indica la finalización del flujo del sistema.
	Decisión: indica un punto en el que el flujo se bifurca.
	Actividad: representa una actividad que se realiza en un proceso.
	Línea de flujo: proporciona el sentido del flujo para un proceso.

Tabla 83: Notación utilizada para realizar el diagrama de flujo.

A continuación se muestra el diagrama de flujo que seguirá la aplicación:

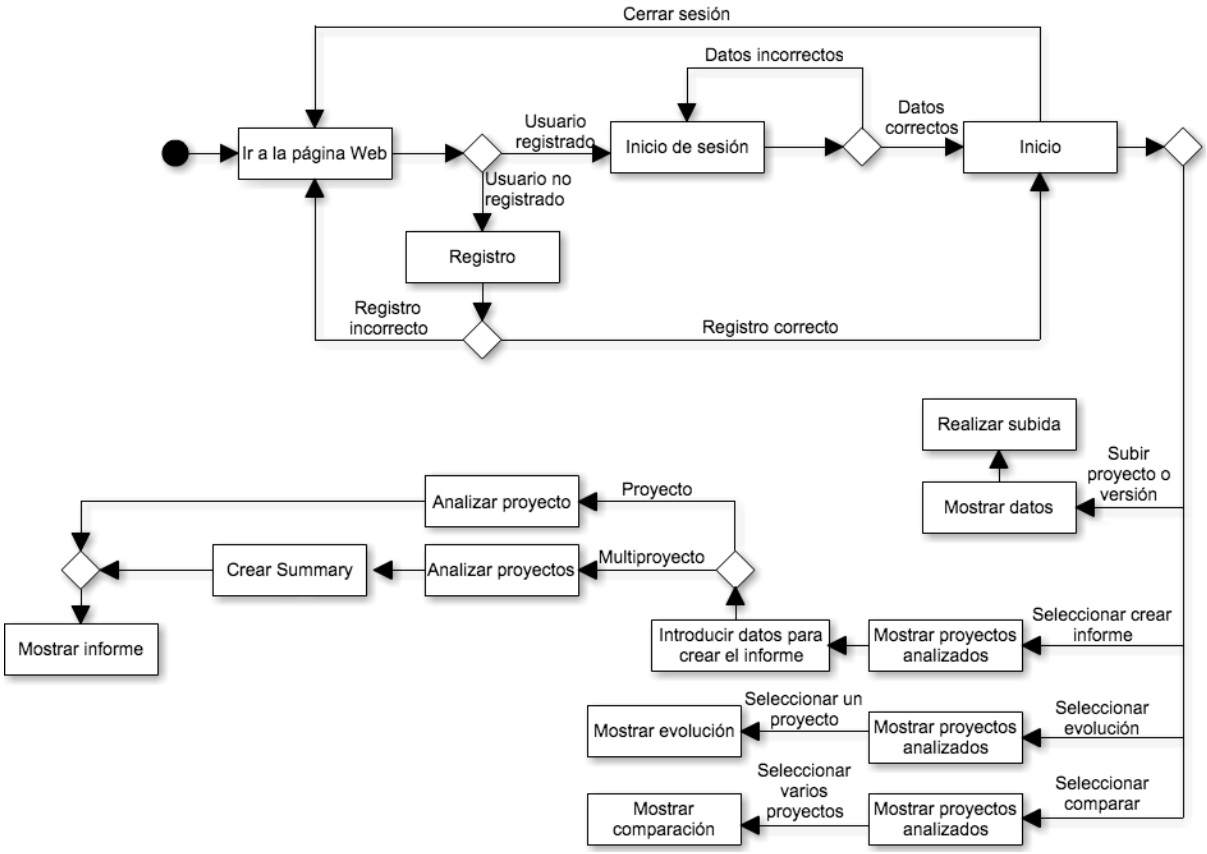


Ilustración 16: Diagrama de flujo del sistema.

4.3. Diagrama de Componentes

En este apartado se realizará una división del sistema en componentes [35], con el fin de describir y representar de forma gráfica todos los componentes que formarán el sistema y las relaciones existentes entre ellos.

El término componente abarca a todos los elementos software de los que se formará el sistema como: archivos, ejecutables, bases de datos, bibliotecas, etc. Esto se traduce en que todo el sistema quedará representado en el diagrama.

La notación que se va a utilizar para denotar a un componente es la que se muestra en la Ilustración 17. En cuanto a los estereotipos, o tipo del componente, UML define los siguientes:

- Executable: representa a todo aquel componente que puede ser ejecutado.
- Library: representa a toda aquella biblioteca de objetos estática o dinámica.
- Table: representa a una tabla de una base de datos.
- File: representa a un archivo que contiene código fuente o datos.
- Document: representa documentos.

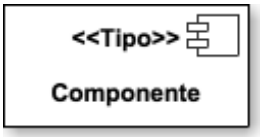


Ilustración 17: Representación de un componente.

Los componentes cuentan con dependencias entre sí, es decir, si un componente necesita alguna funcionalidad o servicio que ofrece otro componente, se denotará mediante una relación de dependencia, tal y como se muestra en Ilustración 18.

Ilustración 18: Representación de dependencias entre componentes.

Por último, se pueden agrupar los componentes en paquetes atendiendo a su criterio lógico para simplificar la posterior implementación. Es importante que un subsistema no sólo puede contener componentes sino que puede contener a su vez otros subsistemas. La representación gráfica de un subsistema es la que se muestra en la Ilustración 19.

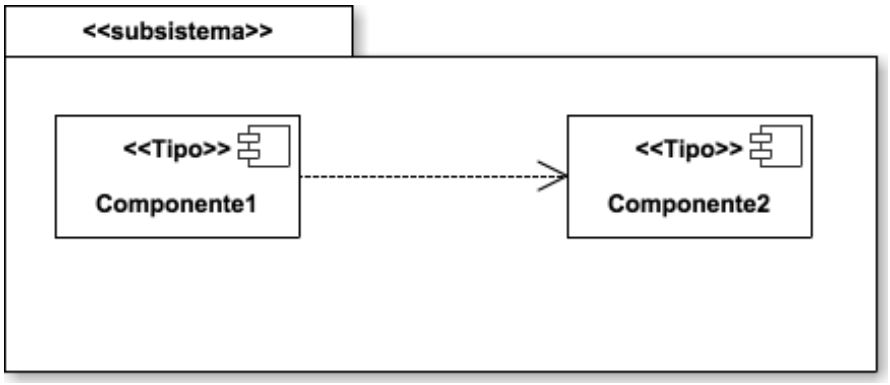


Ilustración 19: Representación de subsistema.

A continuación se presentan los diagramas de secuencia. Es importante destacar que se realizará una división del diagrama de componentes para favorecer la legibilidad. La división se ha realizado mediante funcionalidades, por lo que hay que tener en cuenta que los distintos diagramas pueden dar una sensación de aislamiento cuando en realidad todos los componentes están conectados unos con otros.

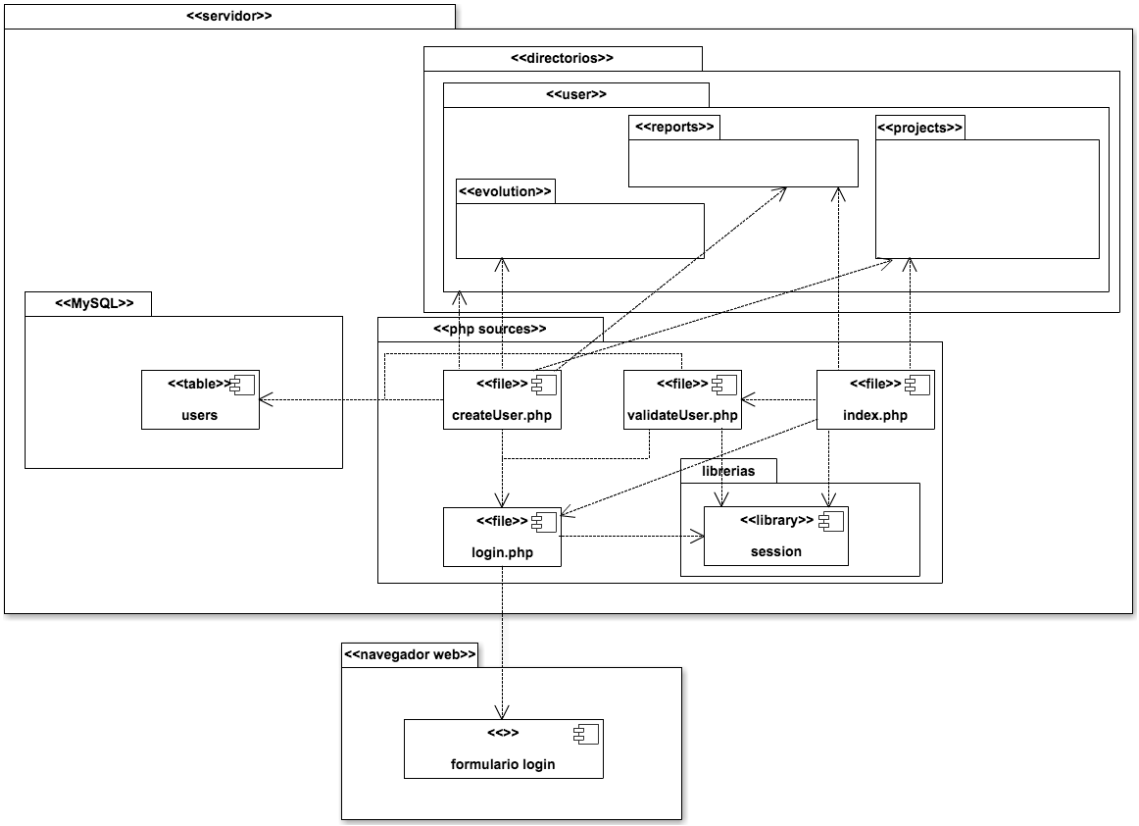


Ilustración 20: Diagrama de componentes para el registro e inicio de sesión en el sistema.

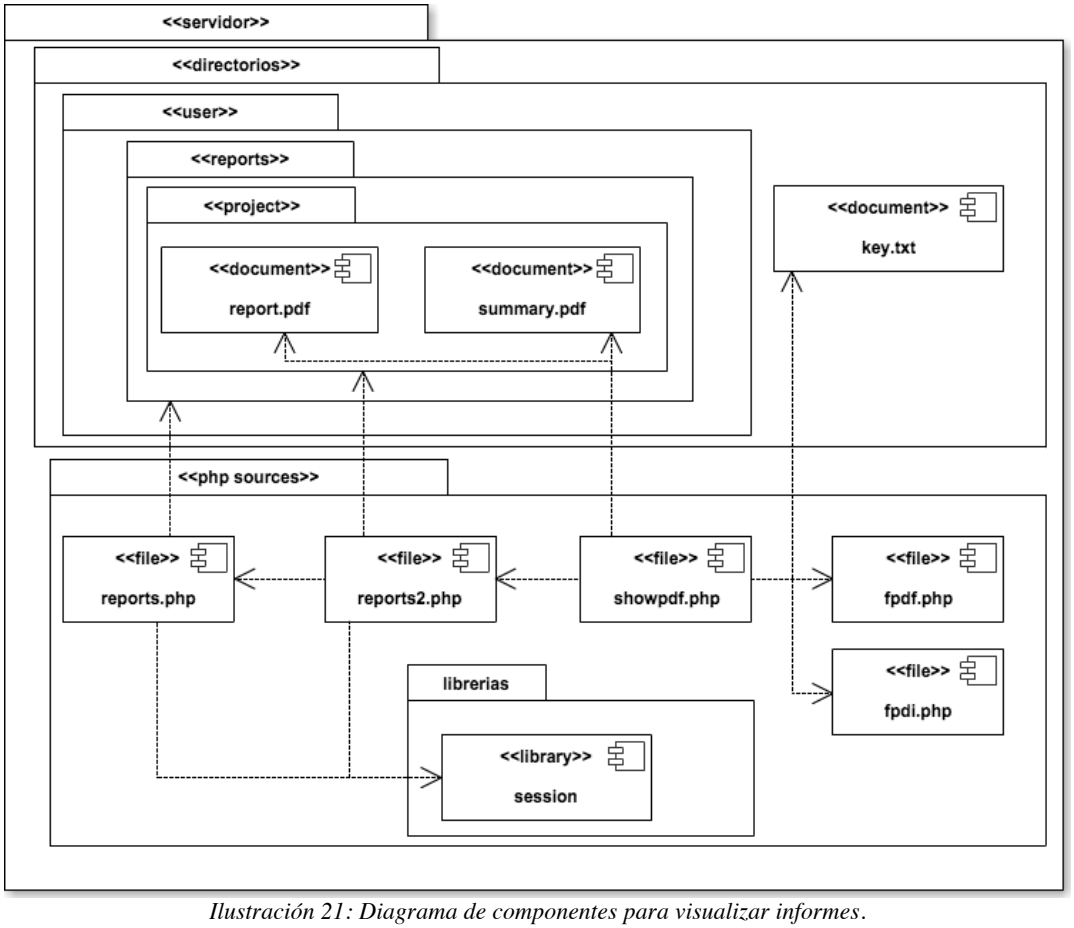


Ilustración 21: Diagrama de componentes para visualizar informes.

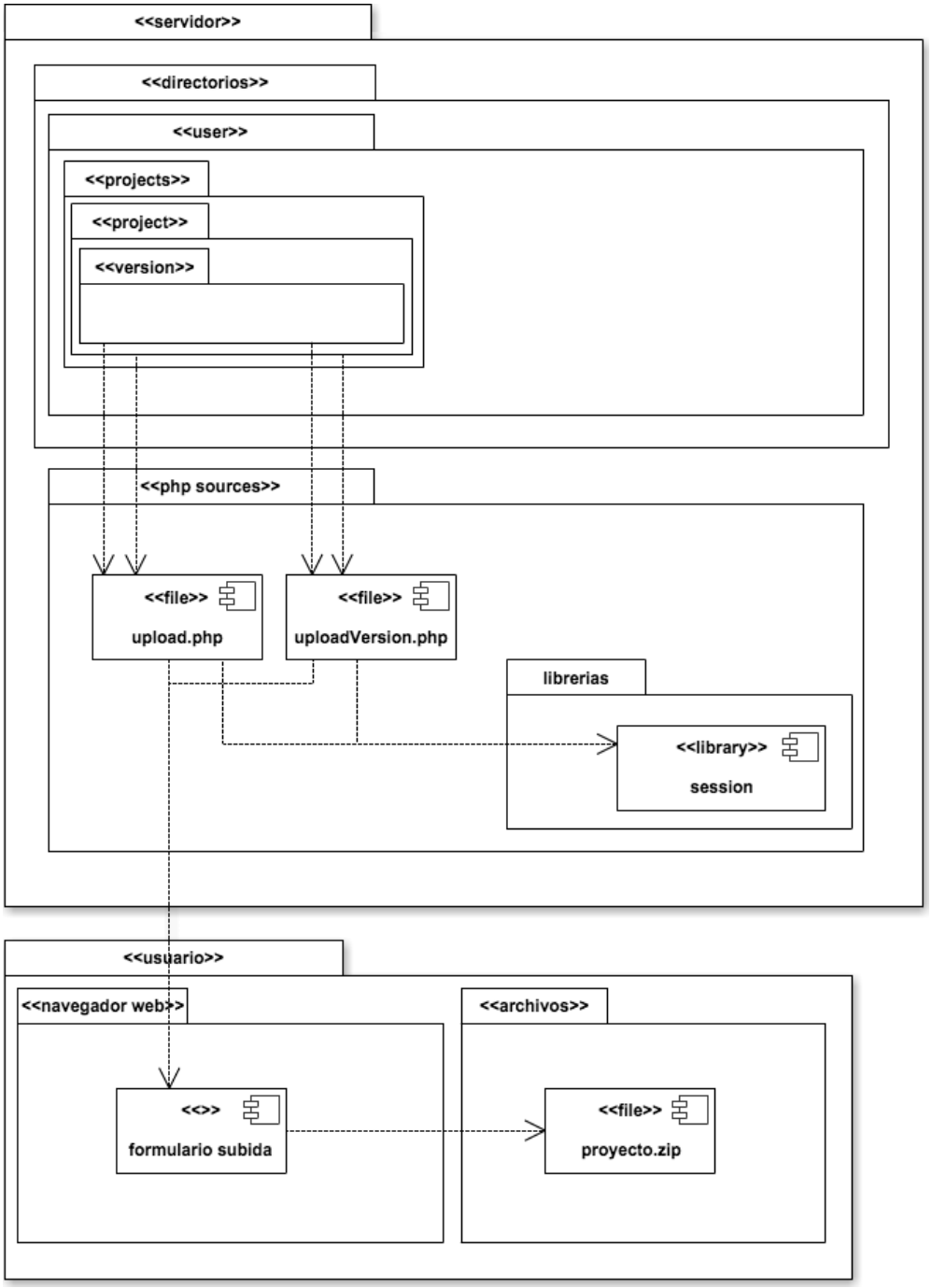


Ilustración 22: Diagrama de componentes para la subida de proyectos y versiones.

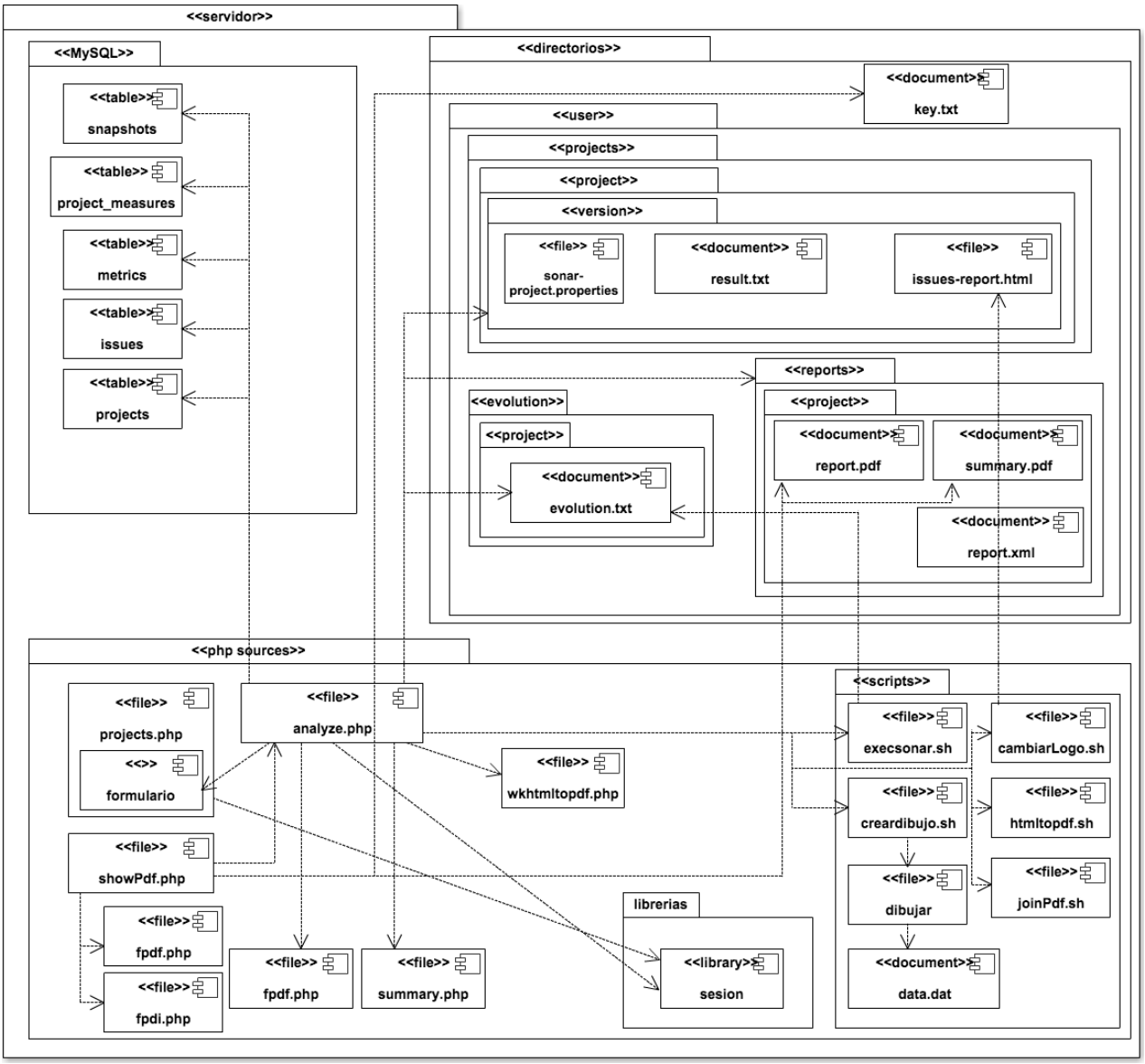


Ilustración 23: Diagrama de componentes para el análisis y creación de informes.

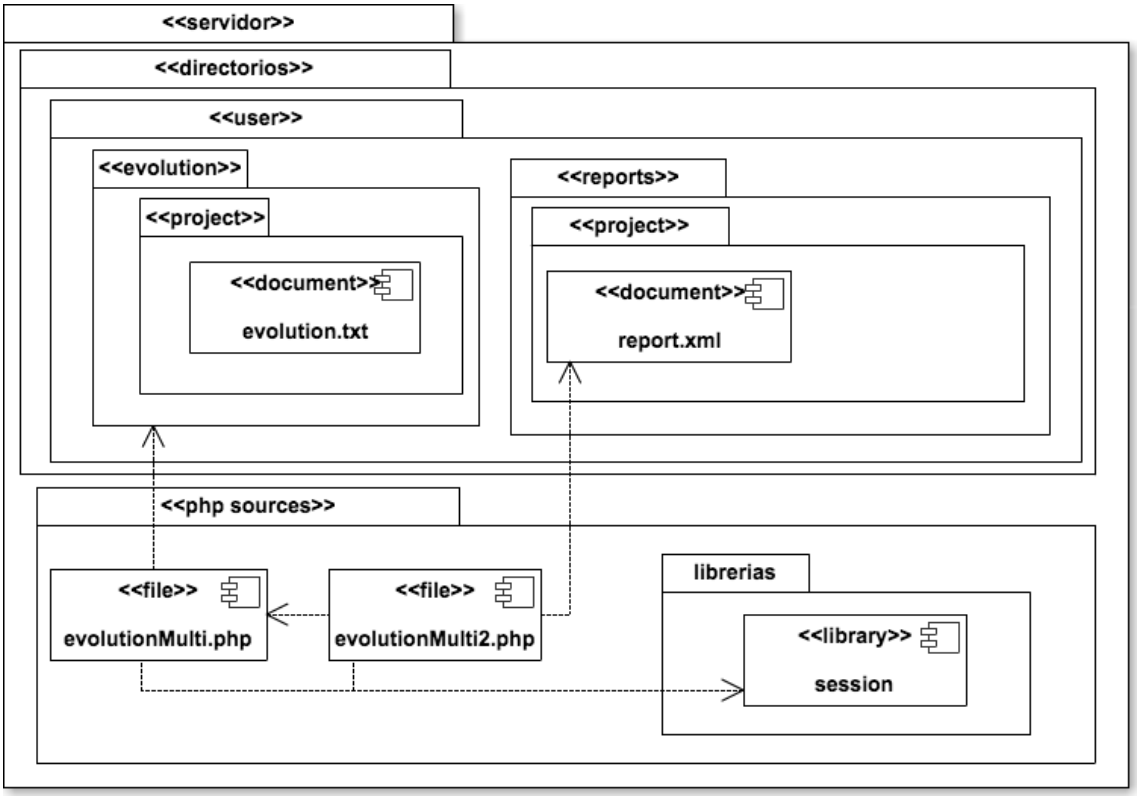


Ilustración 24: Diagrama de componentes para comparar proyectos.

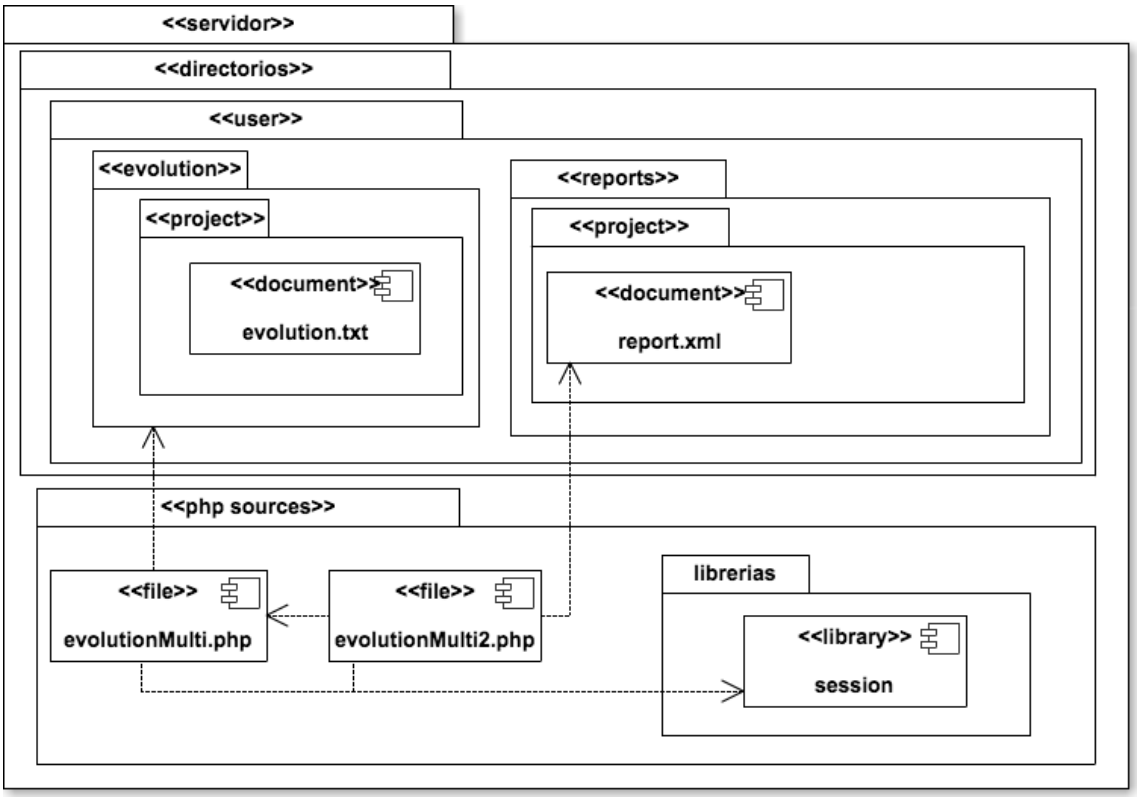


Ilustración 25: Diagrama de componentes para mostrar la evolución de un proyecto.

4.4. DIAGRAMA DE SECUENCIA

En este apartado se mostrarán los distintos diagramas de secuencia derivados directamente de los de componentes. Para no desbordar el contenido de este documento, solo se presentarán los diagramas de secuencia más relevantes.

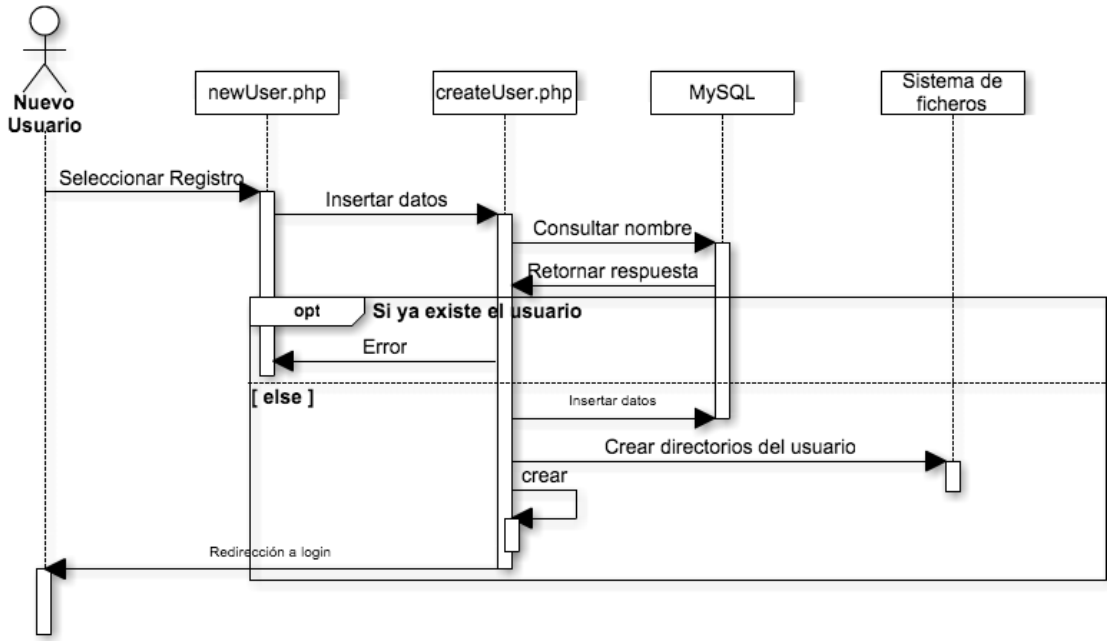


Ilustración 26: Diagrama de secuencia de la actividad de registro.

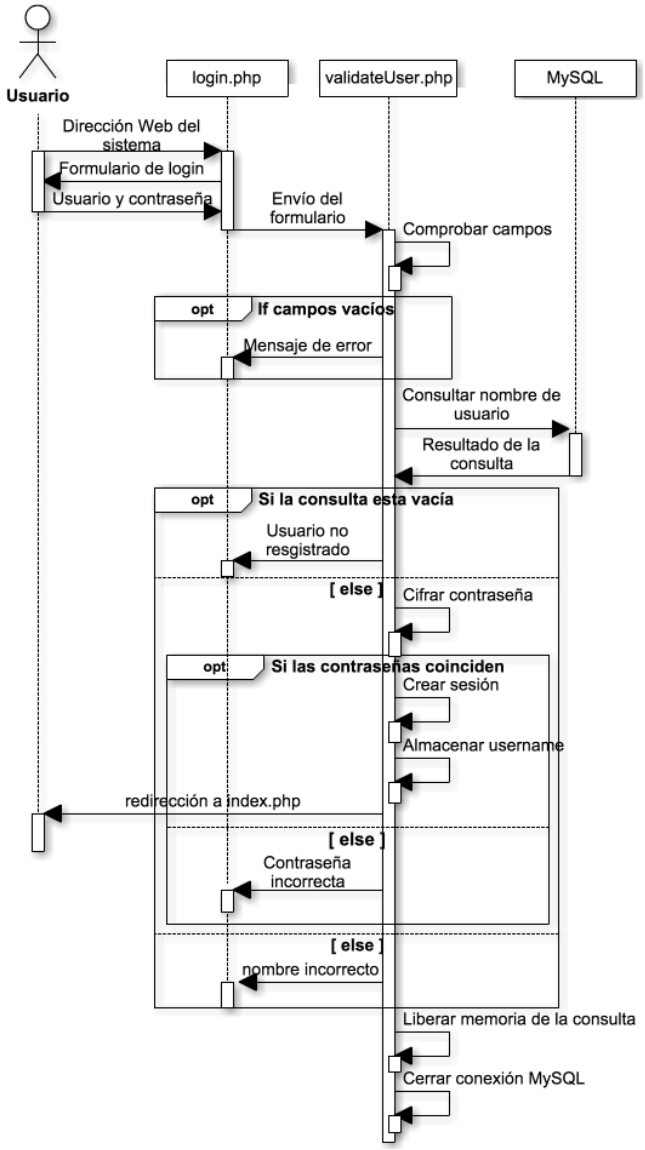


Ilustración 27: Diagrama de secuencia para el inicio de sesión.

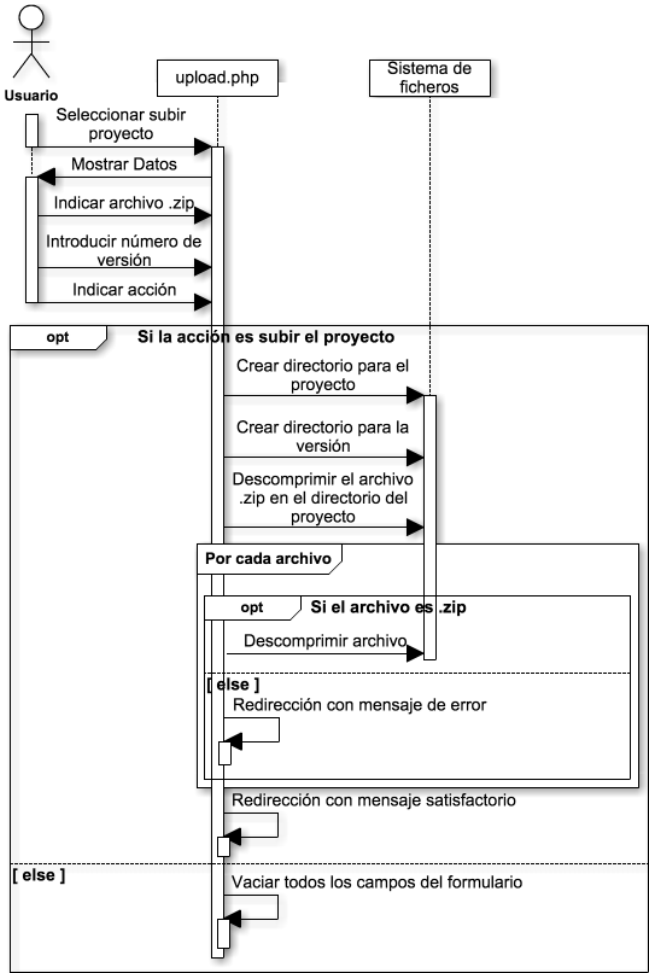


Ilustración 28: Diagrama de secuencia para la subida de un proyecto.

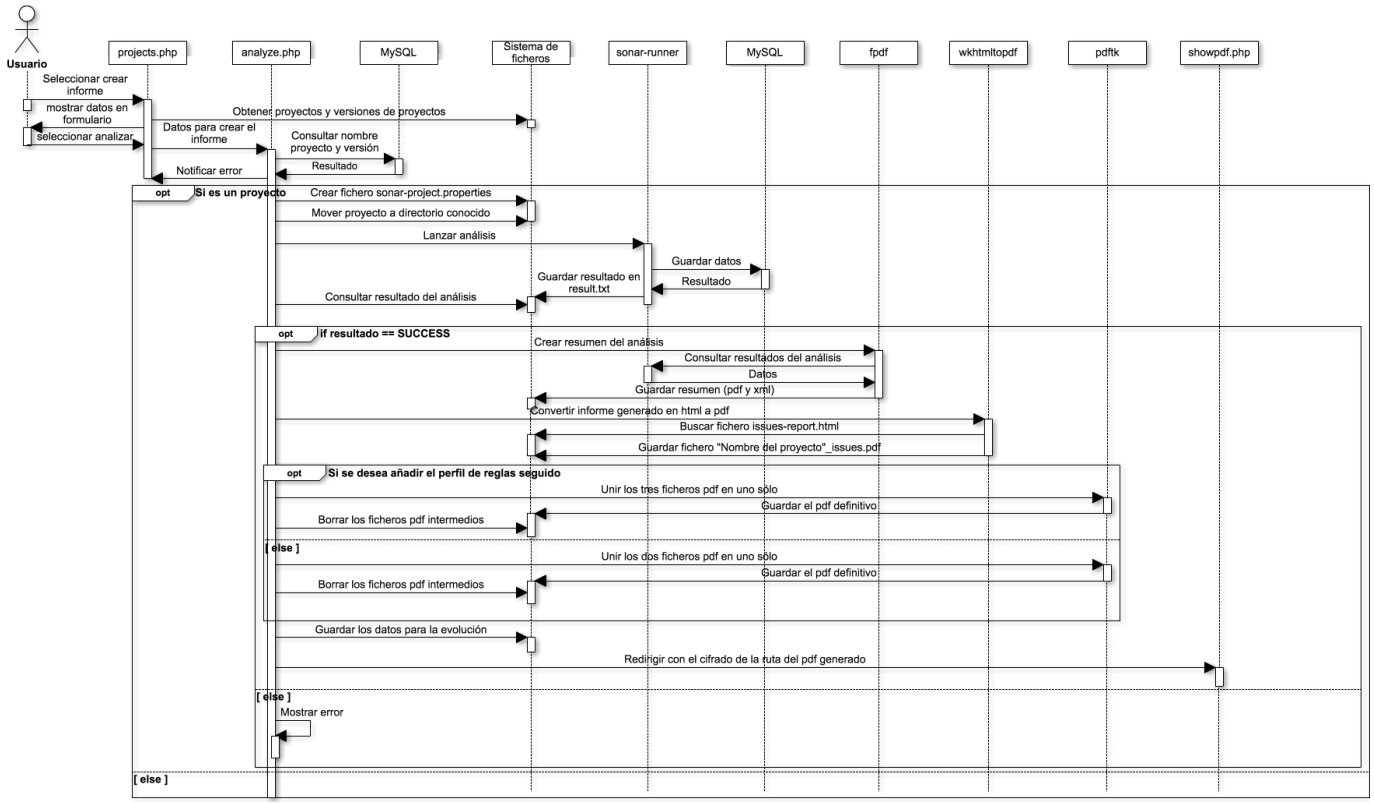


Ilustración 29.1: Diagrama de secuencia de la actividad de análisis (análisis de un sólo proyecto).

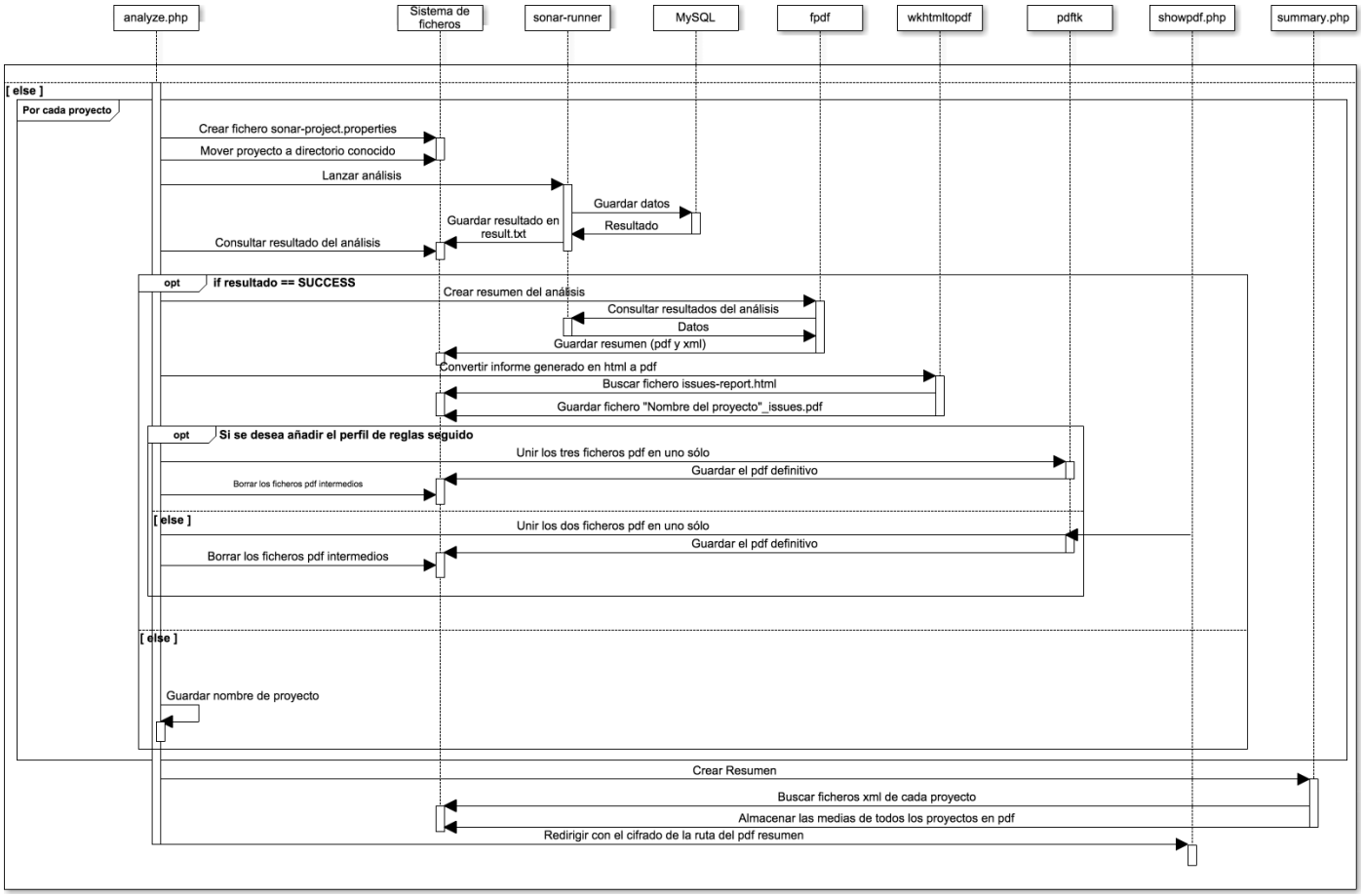


Ilustración 29.2: Diagrama de secuencia de la actividad de análisis (análisis de un multi-proyecto).

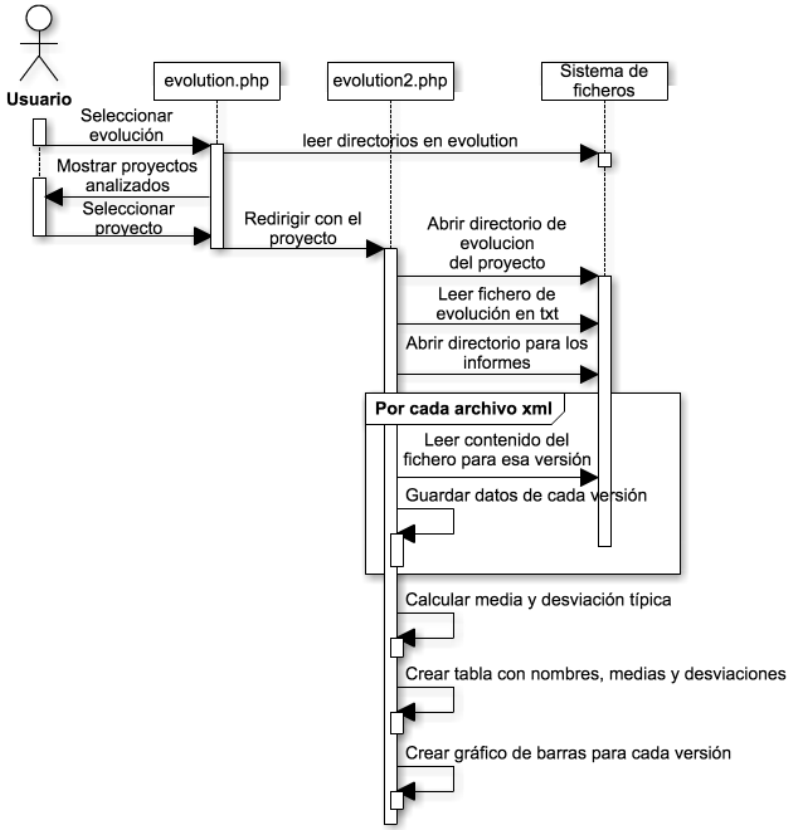


Ilustración 30: Diagrama de secuencia para mostrar la evolución de un proyecto.

4.5. Modelo de Datos

En esta sección se presenta el modelo de datos que se va a utilizar en el sistema. Es muy importante indicar que SonarQube ya proporciona una base de datos, y que no se hará uso de todas las tablas que proporciona. Por este motivo, se presentan a continuación el modelo adicional de tablas de base de datos que se utilizarán para realizar el sistema.

Como se puede comprobar en la Ilustración 31, en la que se presenta el modelo de datos utilizado en el sistema, las tablas no tienen relación entre ellas. Esto es porque el modelo de datos que establece Sonar es no relacional. En cuanto al sistema a desarrollar, no comprometerá la integridad de los datos, ya que realiza consultas sobre las tablas, a excepción de la tabla de usuarios.

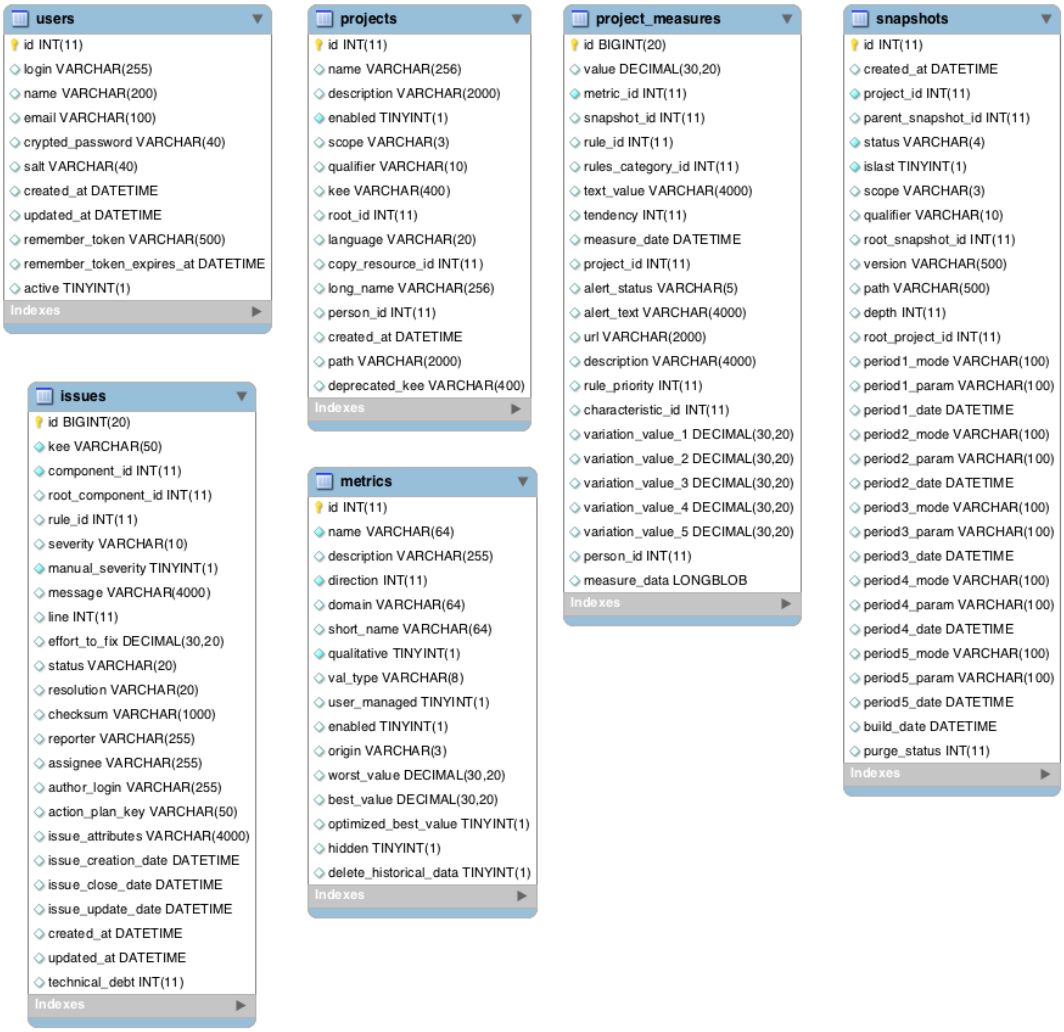


Ilustración 31: Modelo de datos utilizado en el sistema.

4.6. ALTERNATIVA DE DISEÑO

Una alternativa de diseño para este sistema sería reunir los dos servidores, en el que reside el entorno web y el servidor con la base de datos MySQL, en uno solo, tal y como se muestra en la Ilustración 32.

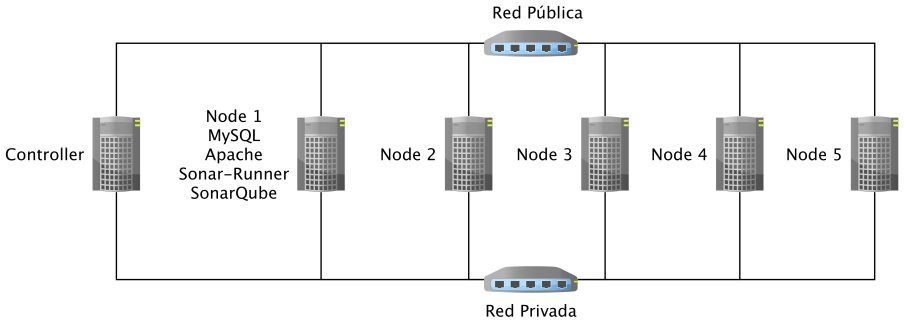


Ilustración 32: Alternativa de diseño para la arquitectura interna del sistema cloud

Si se adquiere este nuevo diseño, es necesario realizar una serie de configuraciones adicionales dentro del servidor, con el objetivo de restringir los accesos que se realicen desde el exterior. Se tienen que realizar las siguientes configuraciones en el cortafuegos que proporciona el sistema de *cloud computing* OpenStack:

- Habilitar los accesos en el puerto 22 por protocolo TCP para dar acceso mediante SSH para poder realizar modificaciones en el servidor.
- Habilitar los accesos por el puerto 9090 por protocolo TCP, con el fin de que los administradores del sistema SonarQube puedan gestionar los perfiles de reglas. Como medida adicional, se puede habilitar el acceso remoto para que sólo ciertas direcciones IP puedan acceder.
- Habilitar accesos por el puerto 443 por el protocolo TCP, para habilitar el acceso seguro mediante HTTPS.

Por último, cabe destacar que no se deberán habilitar ningún tipo de reglas hacia el puerto 3306, ya que es el que utiliza MySQL. De esta forma la información contenida en la base de datos está bloqueada hacia accesos externos.

CAPITULO 5
IMPLEMENTACIÓN

En este capítulo se desarrollan los puntos más importantes de la implementación del sistema. La implementación se ha realizado siguiendo los siguientes pasos:

- Creación de la instancia en el Cloud.
- Instalación del analizador de código.
    - Instalación de SonarQube.
    - Instalación de Sonar-Runner.
    - Instalación y configuración de MySQL.
- Instalación y creación del entorno WEB.
    - Instalación de Apache 2.
    - Instalación de las herramientas para PHP.
    - Programación de la infraestructura.
- Creación de certificados digitales.

5.1. CREACIÓN DE LA INSTANCIA EN EL CLOUD

Para albergar todo el sistema es necesario contar con un computador que actúe como servidor. Este computador es una instancia en el cloud del grupo de investigación ARCOS de la Universidad Carlos III de Madrid. El sistema operativo elegido es Ubuntu 14.04 LTS ya que dispone de 5 años de soporte.

OpenStack permite crear máquinas virtuales a partir de imágenes de sistemas operativos previamente añadidos a su directorio de imágenes, por lo que se generó una máquina siguiendo el sistema habitual de creación. Tras detectar un fallo generado con la máquina que alberga el componente MySQL, el cual se explicará en la sección dedicada a este componente, y de sufrir un corte de luz en las instalaciones donde está el Cloud, la instancia sobre la que estaba desplegando el sistema quedo inservible. Para volver a tener un entorno operativo fue necesario utilizar los numerosas copias de seguridad, y bajar el código fuente del proyecto del repositorio remoto de Bitbucket. Tras esta serie de problemas se decidió realizar otro tipo de instalación.

La instalación definitiva consiste en la creación de un volumen vitual sobre el que se montará todo el sistema de ficheros y el propio sistema operativo. Esta instalación consiste en la creación una instancia a la que se le asocia un volumen y la imagen del sistema operativo. Durante la instalación se monta el volumen y se indica la partición

sobre la que se instalará el sistema operativo. Gracias a este método se consigue tener todo el sistema guardado en un volumen de datos, el cual se replica dentro del Sistema de ficheros Ceph automáticamente. Tras esta instalación se produjo un nuevo corte, y, al igual que en el otro, la instancia se quedo en estado de error, pero se levanto una nueva instancia a partir del volumen creado, por lo que no fue necesaria realizar ninguna configuración y en pocos minutos estaba totalmente operativo.

5.2. INSTALACIÓN DEL ANALIZADOR DE CÓDIGO

Esta sección cubre todos los pasos seguidos para el despliegue del analizar de código sobre la infraestructura desarrollada.

5.2.1. INSTALACIÓN DE SONARQUBE

El componente encargado de realizar el análisis de código en este proyecto será SonarQube, su instalación [36] consiste básicamente en descargar la distribución deseada, en este caso la versión 4.5.2, descomprimirla en una ruta cualquiera, `/etc/sonar/`, y lanzar el comando `/etc/sonar/bin/linux-x86-64/sonar.sh start`" para iniciar el servidor propio que implementa. Esta instalación cuenta con una particularidad, y es que obliga en cada reinicio de la instancia a la ejecución del comando de arranque de SonarQube. Por este motivo, es necesario realizar una instalación de SonarQube como servicio propio de Ubuntu. Para realizar la instalación de este componente como servicio [37] basta con crear un archivo llamado `sonar` dentro del directorio `init.d` con el siguiente contenido:

```
#!/bin/sh
#
# rc file for SonarQube
#
# chkconfig: 345 96 10
# description: SonarQube system (www.sonarsource.org)
#
### BEGIN INIT INFO
# Provides: sonar
# Required-Start: $network
# Required-Stop: $network
# Default-Start: 3 4 5
# Default-Stop: 0 1 2 6
# Short-Description: SonarQube system
(www.sonarsource.org)
# Description: SonarQube system (www.sonarsource.org)
### END INIT INFO
/etc/sonar $*
```

Además, es necesario registrar el servicio dentro del arranque del Sistema Operativo Ubuntu, para ello se lanzan los siguientes comandos en la terminal:

```
sudo ln -s /etc/sonar/bin/linux-x86-64/sonar.sh /etc/sonar /bin/sonar
sudo chmod 755 /etc/init.d/sonar
sudo chkconfig --add sonar
```

SonarQube proporciona un entorno propio web, el cual se utilizará como entorno de administración en este proyecto.

5.2.2. Instalación de Sonar-Runner

Una vez instalado SonarQube, es necesario instalar la parte que se encarga de realizar el análisis de los proyectos. Para ello, SonarQube proporciona un analizador propio llamado Sonar-Runner [38]. Para instalar este componente basta con descargar el analizador, situarlo en un directorio personalizado, `/etc/sonar-runner` y configurar el analizador. Para realizar la configuración es necesario modificar el fichero `sonar-runner.properties` situado en el directorio `/etc/sonar-runner/conf/` con los siguientes datos:

```
#----- Default SonarQube server
sonar.host.url=https://localhost:9090/sonar

#----- MySQL
sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?useUnicode=true&amp;c
haracterEncoding=utf8

#----- Global database settings
sonar.jdbc.username=**********
sonar.jdbc.password=**********

#----- Default source code encoding
sonar.sourceEncoding=UTF-8
```

Como se puede apreciar en el extracto del fichero de configuración, el nombre de usuario y la contraseña han sido ocultadas para conservar la seguridad y la integridad del Sistema.

5.2.3. Instalación de MySQL

Por último, para dejar el sistema de análisis al completo, falta por instalar la base de datos MySQL. Una vez instalada, solamente queda por ejecutar el siguiente script [39]:

```
CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8_general_ci;
CREATE USER 'sonar' IDENTIFIED BY 'sonar';
GRANT ALL ON sonar.* TO 'sonar'@'%' IDENTIFIED BY 'sonar';
GRANT ALL ON sonar.* TO 'sonar'@'localhost' IDENTIFIED BY 'sonar';
FLUSH PRIVILEGES;
```

Es importante destacar que la base de datos produjo un error durante los cortes de corriente sufridos en el Cloud. El error en cuestión era que no se podía conectar a la base de datos de forma local porque se perdía una referencia con el archivo `mysql.sock`. Sobre este error existen muchos hilos abiertos en internet, pero ninguno ofrece una solución sencilla y aun menos rápida. La solución adoptada en este proyecto fue la de instalar el componente MySQL en una máquina aparte y sobre un volumen de datos. De esta forma se consigue una replicación automática y no se pierde la conexión en caso de cortes de electricidad.

Después de realizar estos pasos, el sistema cuenta con la parte de análisis requerida para realizar los informes. Para poder realizar un análisis [40] sobre un proyecto, basta con tener el código fuente del proyecto y crear un archivo con el nombre `sonar-project.properties` con el siguiente contenido:

```
sonar.projectKey=”Nombre del proyecto”
sonar.projectName=”Nombre del proyecto”
sonar.projectVersion=”Versión del proyecto”
sonar.sources=.
sonar.login=********
sonar.password=********
sonar.profile=”Perfil de reglas”
sonar.language=”Lenguaje del proyecto”
```

Para que Sonar realice un análisis sobre el proyecto es necesario que se ejecute el siguiente comando: `/etc/sonar-runner/bin/sonar-runner`. Tras lanzar el comando, el proyecto será analizado y se almacenarán los resultados en la base de datos.

5.3. Instalación y creación del entorno WEB

En esta sección se ofrece una descripción sencilla de los pasos más destacados durante la implementación del Sistema Web, el cual utilizarán los usuarios finales.

5.3.1. Instalación de Apache2

Para poder ofrecer al usuario un entorno online, es necesario instalar un servidor web para alojar el sistema presentado. Como ya se ha comentado anteriormente, el servidor web elegido es Apache 2. Para instalarlo, basta con actualizar primero el sistema operativo mediante el comando `apt-get update`. A continuación, se realiza la

instalación del servidor mediante el siguiente comando: `apt-get install apache2`. Tras la instalación se puede verificar que se ha instalado correctamente introduciendo el enlace a la máquina en cualquier navegador web, en este caso la dirección IP de la máquina es `163.117.148.119`.

5.3.2. Instalación de las herramientas para PHP

Para que el servidor web sea capaz de interpretar las páginas escritas en código PHP, es necesario instalar los siguientes componentes de PHP:

- Componente php5: se trata de todas las librerías necesarias para poder generar y compilar código en PHP. Además, permitirá que el sistema pueda ejecutar código PHP como scripts de la Shell de Ubuntu.
- Componente libapache2-mod-php5: contiene todas las librerías para configurar de forma automática PHP con Apache2. De esta forma, las páginas codificadas en PHP se compilarán en el servidor, generando su código HTML equivalente, el cual se les mostrara a los usuarios en el navegador siguiendo el protocolo de transferencia HTTP.

El comando necesario para instalar estos componentes es: `apt-get install php5 libapache2-mod-php5`. Tras la instalación de este componente es conveniente reiniciar el servidor Apache para que los cambios realizados tengan su efecto.

5.3.3. Programación de la infraestructura

5.3.3.1. Acceso y registro

Este componente representa una de las secciones críticas del proyecto, ya que representa la primera barrera por la que tienen que pasar los usuarios. Los archivos PHP necesarios son los siguientes:

- Archivo `login.php`: este archivo se compone básicamente de dos campos de texto (nombre de usuario y contraseña), el botón Login y una redirección a la página `createUser.php`. El botón Login, al ser pulsado, envía el formulario a la página validateUser.php.
- Archivo `validateUser.php`: este archivo recibe el nombre de usuario y contraseña del formulario que envía la página `login.php`. Primero, se comprueba que estos campos no estén vacíos. Una vez realizadas las comprobaciones básicas, se recupera de la base de datos la fila que contiene ese nombre de usuario. Si el resultado de la consulta es nulo, quiere decir que el usuario no existe en el sistema. Si se consigue una fila, se compara el hash de la contraseña con el campo contraseña de la fila obtenida. Si no concuerdan es que

la contraseña introducida no es correcta. Cada error producido conlleva una redirección a la página `login.php` con un mensaje de error. Si los campos introducidos son correctos, se crea la sesión del usuario, se inserta en ella el nombre del usuario y se realiza una redirección a la página `login.php`, por lo que se le brinda el acceso al usuario.

- Archivo `newUser.php`: esta página se compone de un formulario con los campos nombre, usuario, contraseña, confirmación de contraseña y el botón de envío de formulario. El formulario se envía a la página `createUser.php`.
- Archivo `createUser.php`: este fichero PHP es el encargado del registro de usuarios. Para ello primero se obtienen todos los datos del formulario que envía la página `newUser.php`. Después, se comprueba que ambas contraseñas concuerdan. Si no concuerdan las contraseñas, se notifica con un error. Si concuerdan, se consulta a la base de datos si el usuario existe en el sistema. Si no existe, se insertan todos los valores introducidos por el usuario en la base de datos, se crea la sesión y los directorios privados del usuario en el sistema. Por último, se realiza una redirección a la página `index.php`.

5.3.3.2. Subida de archivos

Este componente ofrece a los usuarios la subida de código al sistema. Para realizar la subida, el código debe estar comprimido en un fichero .zip. Además, se distinguen dos tipos de subidas. Por una parte se encuentra la subida del código de nuevos proyectos, y por otra la subida de nuevas versiones de código para proyectos que ya están creados en el sistema.

Para implementar este mecanismo de subida se necesitan los siguientes archivos:

- Archivo `upload.php`: este archivo cuenta con un formulario con los siguientes componentes: entrada numérica para la versión, entrada para un fichero y el botón de envío de formulario. El campo de versión es decimal con un valor mínimo de 0 y cuenta con incrementos de 0,01. Cuando se realiza el envío del formulario, la página de destino es ella misma, ya que al comienzo del archivo se realizan las siguientes operaciones: Primero, se obtiene el fichero comprimido y se comprueba que su codificación sea .zip. Si su codificación es la deseada, se crea el directorio del proyecto y dentro de este un directorio con el nombre de la versión. Dentro del directorio de la versión se realiza la descompresión del código. Una vez descomprimido, se revisan todos los archivos por si existen ficheros comprimidos, ya que se contempla la subida de un proyecto compuesto por varios proyectos, destinado al análisis de prácticas, etc. En caso de que se produzca algún error, se notifica al usuario en esta misma página.
- Archivo `uploadVersion.php`: este archivo es muy parecido a `upload.php`, con la salvedad de que en el formulario se añade un selector en el que se muestran todos los proyectos del usuario, ya que la versión a subir pertenece a

uno de ellos. El campo versión es decimal con un valor mínimo de 0 y cuenta con incrementos de 0,01. El formulario se envía a sí misma y se realizan las siguientes operaciones: Primero, se obtiene el fichero comprimido y se comprueba que su codificación sea .zip. Si su codificación es la deseada, se crea el directorio de la versión dentro del directorio del proyecto. Dentro del directorio creado se realiza la descompresión del código. Una vez descomprimido, se revisan todos los archivos por si existen ficheros comprimidos. Si existe se hace una descompresión recursiva.

Cabe destacar un error detectado durante la implementación de este componente. El fallo consiste en tamaño máximo por defecto que permite PHP para los archivos de subida. Por defecto permite la subida de archivos de unos pocos Megabytes, si el fichero con el código supera este valor, se aborta la subida. Para solucionarlo es necesario cambiar la configuración por defecto establecida en el fichero `/etc/php5.ini`. En este fichero se establece el tamaño máximo de los archivos que se van a subir a 6 Gigabytes.

5.3.3.3. Creación de informes

Este componente representa la parte crítica del sistema, ya que en él se realiza la funcionalidad de analizar el código fuente y crear los informes con los datos obtenidos. Los archivos que lo componen son muy variados, tanto en funcionalidad como en lenguajes de programación:

- Archivo `projects.php`: se compone de un formulario con cuatro selectores, dos checkbox y el botón de envío del formulario. De los selectores hay que destacar que dos de ellos se cargan dinámicamente dependiendo del valor seleccionado en otro, es decir, el selector de versiones carga todas las versiones subidas para el proyecto seleccionado, y, el selector de perfiles carga todos los perfiles que corresponden al lenguaje seleccionado anteriormente. Además, los dos check aportan al usuario la funcionalidad de añadir un anexo al final del informe generado y la opción de realizar un análisis de varios proyectos contenidos en uno solo. Por último, cabe destacar que el botón de envío del formulario tiene como destino el archivo `analyze.php`.
- Archivo `analyze.php`: este archivo contiene solamente código PHP, se encarga de realizar el análisis y creación de informes de proyectos. Para ello realiza las siguientes funciones:
    - Obtener todos los datos del formulario y comprobar que todos ellos han sido seleccionados. En caso de que algún campo no haya sido seleccionado, se notifica el aviso y se realiza una redirección a `projetcs.php`. Además, se comprueba que Sonar no haya realizado un análisis sobre la versión del proyecto seleccionada anteriormente.
    - Una vez analizados todos los datos introducidos por el usuario, se comprueba si es un análisis múltiple, analizar varios proyectos dentro de

uno, o un análisis individual. Para el caso simple, se realiza la creación del fichero `sonar-project.properties`. Una vez creado el archivo necesario para que SonarQube realice el análisis, se realiza una llamada al script execSonar.sh, el cual lanza el análisis y guarda el resultado en un fichero llamado result.txt. A continuación, se comprueba que el análisis ha sido satisfactorio leyendo el fichero con el resultado. Si el análisis no se ha podido realizar, se le notifica al usuario, en caso contrario, se crea el informe en XML. Este informe será utilizado internamente por el sistema, mediante la librería nativa de PHP, un fichero con formato de texto plano con datos para seguir la evolución del proyecto, evolution.txt, y el informe en PDF. Para la creación del PDF se distinguen dos pasos:

1. Creación de la primera hoja con los datos globales del proyecto: nombre, versión, y una hoja de resumen, la cual se extrae de los datos almacenados en MySQL. Es importante indicar que algunos de los datos mostrados en el resumen son gráficos. Para su creación se realiza el componente GNUPlot, el cual genera los gráficos mediante el script crearDibujo.sh.
2. Convertir el informe que genera SonarQube en formato HTML en un fichero PDF mediante el script htmltopdf.sh. Este informe contiene todos los errores que contiene el código junto con las partes del código donde se da el error.

Una vez generados estos dos ficheros PDF, es necesario unirlos en uno sólo mediante la utilización del script `joinPdf.sh`. Tras la unión de los ficheros en uno sólo, se realiza la visualización del PDF resultante.

Si el análisis que se debe realizar es múltiple, el funcionamiento es idéntico al descrito anteriormente, con la diferencia en la creación de un fichero PDF con las medias de todos los proyectos, es decir, el sistema analiza todos los proyectos de los que se compone, y crearán sus correspondientes ficheros PDF y XML. A partir de los ficheros XML se extraen todos los datos y se realiza la media de todos los valores generando a su vez un PDF que contiene todos los datos estadísticos.

- Script `execSonar.sh`: este Script recibe la ruta en la que está almacenado el proyecto y la ruta en la que se realizará el análisis. Después, lanza el análisis del proyecto y redirige la salida hacia el fichero `result.txt`. Por último, copia todos los ficheros que genera Sonar bajo un directorio oculto.
- Script `joinPdf.sh`: este Script recibe la ruta de los ficheros PDF que se deben unir y el directorio en el que se almacenará el resultado. Después de realizar la unión, elimina los archivos originales.
- Script `htmltopdf.sh`: recibe como parámetros las rutas del fichero HTML y del PDF a crear.

- Script `crearDibujo.sh`: el fichero recibe el directorio donde reside el proyecto y el nombre de la imagen que debe crear. Primero, este script crea un archivo con el código ejecutable de GNUPlot, ya que el fichero data.dat es creado por `analyze.php`. Después, se lanza el script creado de GNUPlot, el cual da como resultado una imagen PNG con un nombre determinado. Por último, se elimina el Script de creación y el que contiene los datos.

Durante el desarrollo de este componente se produjo un fallo muy importante. El fallo se detectó al realizar el análisis de un proyecto muy grande, su tamaño superaba los 3 Gigabytes de datos y contaba con mas de 2000 archivos, y consistía en fallos producidos por los nombres de los archivos. Si un archivo contiene espacios, el analizador de código de Sonar falla. Por lo general los compiladores no permiten que los archivos con el código posean espacios entre caracteres, pero sí permiten ficheros de documentación con ese tipo de caracteres. Para solucionarlo, se creó un nuevo script que borra de forma recursiva todos los ficheros que no son de código fuente o librerías. De esta forma se consigue solventar el error y disminuir el tamaño de los proyectos dentro del sistema.

5.3.3.4. Visualización de informes

Este componente es el encargado de visualizar los informes PDF generados por el sistema. Los archivos que forman este componente son los siguientes:

- Archivo `reports.php`: esta página realiza un listado del directorio de informes del usuario. Por cada uno de los proyectos en el directorio se crea un enlace a la página `reports2.php`.
- Archivo `reports2.php`: en esta página primero se comprueba que se ha seleccionado un proyecto. Después, se listan todos los ficheros PDF y, por cada uno, se genera un enlace con la ruta del PDF hacia la página `showpdf.php`. Hay que destacar que la ruta se cifra para ocultar la ruta del archivo a los usuarios.
- Archivo `showpdf.php`: primero se descifra la ruta del archivo que hay que mostrar. Después, mediante la librería fpdi se muestra el fichero en el explorador.

5.3.3.5. Mostrar evolución de un proyecto

Este componente es el encargado de mostrar a los usuarios la evolución de un proyecto a lo largo del proceso de desarrollo, ya que muestra como varían los datos más relevantes de las distintas versiones que analiza. Los archivos que forman el componente son los siguientes:

- Archivo `evolution.php`: esta página realiza un listado del directorio de evolución del usuario. Por cada uno de los proyectos en el directorio se crea un enlace a la página `evolution2.php`.
- Archivo `evolution2.php`: en esta página se abren todos los ficheros XML de las versiones analizadas. De cada fichero se extraen los datos de complejidad y se almacenan en una matriz. Cuando se han leído todos los ficheros se calcula la media y la desviación típica para cada fila de la matriz. Después se muestran los datos en una tabla. Por último, para mostrar el gráfico de barras se hace uso de la librería Google Chart con los datos del fichero de evolución privado para cada proyecto.

5.4. CREACIÓN DE CERTIFICADOS DIGITALES

Como se indica en los requisitos del sistema, para asegurar la seguridad de las conexiones de los usuarios es necesario utilizar el protocolo HTTPS. Para habilitar el protocolo es necesario realizar las siguientes operaciones [41]:

- Generar un certificado digital con OpenSSL.
- Copiar el certificado generado en la ubicación `/etc/sonar/conf`, y configurar el fichero de configuración de Sonar, `/etc/sonar/conf/sonar.properties`, de la siguiente forma:

```
sonar.web.port=-1
sonar.web.https.port=9090
sonar.web.https.keyAlias=*************
sonar.web.https.keyPass=*************
sonar.web.https.keystoreFile=/etc/sonar/conf/server.p12
```

  Con esos parámetros se indica que no se acepten peticiones mediante el protocolo HTTP y si por el protocolo HTTPS por el puerto 9090. Para preservar la seguridad del sistema, no se muestra ni el *keyAlias* ni la contraseña.
- Copiar la llave y el certificado en los directorios `/etc/ssl/private` y `/etc/ssl/certs` respectivamente.
- Modificar el fichero `/etc/apache2/site-available/default-ssl` con los siguientes datos:
  - SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
  - SSLCertificateFile /etc/ssl/certs/server.crt
  - SSLCertificateKeyFile /etc/ssl/private/server.key
- Lanzar el comando: `sudo a2ensite default-ssl`.
- Reiniciar el servidor Apache.

CAPITULO 6
AMPLIACIÓN DE REGLAS PARA SONARQUBE

Una de las herramientas que ofrece SonarQube es la de ampliar el número de reglas para el análisis estático del código. Las formas que existen para agregar las reglas son las siguientes:

- Escribir las reglas en Java a través de un *plugin* de Sonar.
- Añadir reglas XPath mediante la interfaz web ofrecida.

En este capítulo se presenta una guía que facilita el proceso de incorporar nuevas reglas para sonar mediante la interfaz web. De esta forma se analizará el árbol de sintaxis abstracto, el lenguaje de navegación por XML, XPath, y una herramienta de ayuda al análisis de árboles sintácticos llamada SSLR toolkit.

6.1. ÁRBOL DE SINTAXIS ABSTRACTO

El árbol sintáctico nos revela el contenido léxico y la estructura sintáctica de cualquier código fuente en forma de árbol. Cada uno de los nodos representan constantes o variables y los nodos internos representan operaciones y declaraciones.

En cuanto a la sintaxis que sigue este tipo de árboles se considera abstracta, en el sentido en que no muestra todos los detalles que aparecen en la sintaxis verdadera.

```
int main()
{
    int a = 3+1;

   return a;
}
```

Ilustración 33: Código de ejemplo.

Para el fragmento de código presentado en la Ilustración 33, su correspondiente árbol sintáctico queda presentado en la Ilustración 34.

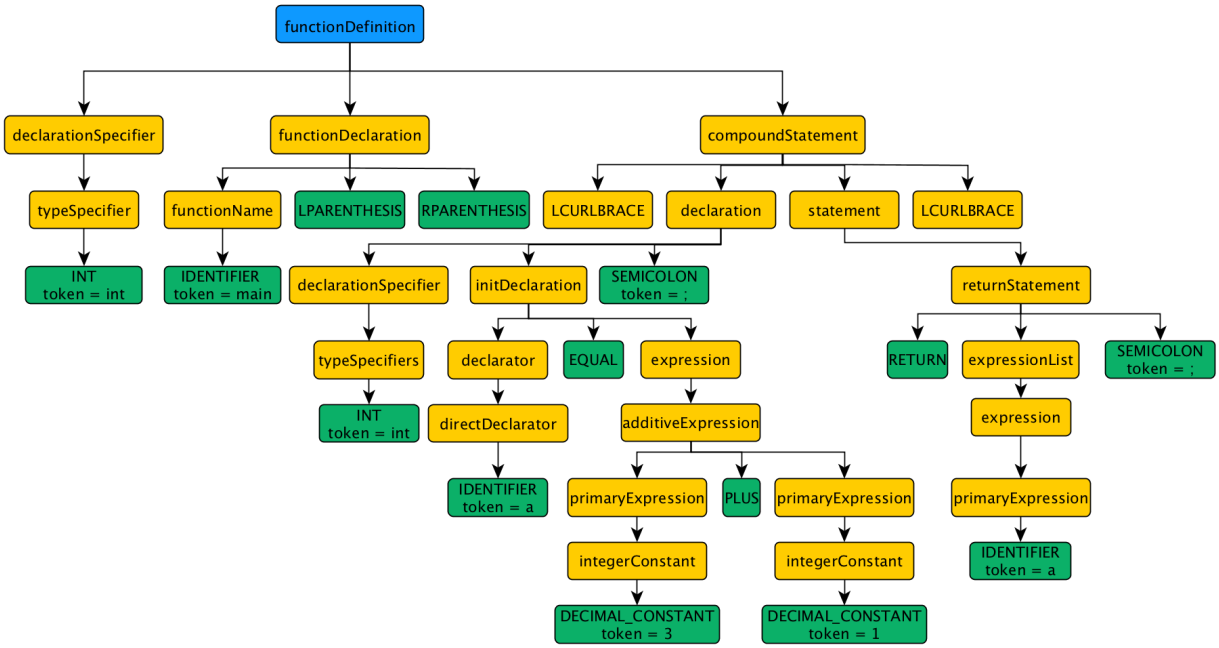


Ilustración 34: Ejemplo de árbol sintáctico.

6.2. XML Path Language

El lenguaje XPath permite construir expresiones para recorrer y procesar documentos XML. La gran ventaja que ofrece XPath es que es muy eficaz a la hora de realizar búsquedas y seleccionar elementos teniendo en cuenta la estructura jerárquica.

Un archivo XML se procesa mediante un analizador y se construye un árbol de nodos, dicho árbol siempre comienza por un elemento raíz que se separa en elementos terminando en nodos hoja que contienen texto, comentarios, instrucciones de proceso y atributos.

6.2.1. Tipos de nodos

En un árbol sintáctico existen varios tipos de nodos:

- Nodo raíz. El nodo raíz es denotado por / y es el origen desde el cual cuelgan todos los elementos del árbol.
- Nodo elemento. Cada uno de los elementos del XML pasa a ser un nodo elemento tras la etapa de procesamiento o parser. Cada uno de estos nodos tiene un nodo padre, a excepción del nodo raíz. También tienen a su vez nodos hijos que son aquellos que contienen texto, comentarios, etc. Todos los nodos elemento poseen propiedades como nombre, atributos e información del espacio de nombres.
- Nodo atributo. Son nodos descendientes de nodos elementos. Estos nodos constan de un nombre y un valor asociado.

- Nodo texto. Este tipo de nodos no tienen hijos, y representan todos aquellos elementos del documento que no están marcados con una etiqueta.

6.2.2. *Location Paths*

Los *Location Paths* son expresiones en XPath que permiten incluir diversas operaciones sobre varios tipos de operandos. Este tipo de expresiones es el más utilizado y su notación es igual a la que se utiliza para recorrer los directorios que forman la unidad de disco en un sistema UNIX.

Cabe destacar que una expresión *Location Path* retorna una referencia a los elementos que cumplen con el patrón presentado, es decir, retorna una lista con aquellos elementos que tienen el mismo patrón que el mostrado con la expresión *Location Path*. La lista retornada puede estar vacía o contener uno o más nodos.

Todo *Location Path* cuenta con los siguientes elementos:

- Nodo contexto. Denota el primer nodo del árbol de la expresión, por lo que es el punto de partida de la expresión.
- Nodo actual. Es el nodo situado al final de la expresión, es decir, denota a todos aquellos nodos que serán retornados por la expresión.
- Predicados. Están incluidos dentro de las expresiones mediante corchetes, y sirven para seleccionar aquellos nodos que cumplen con un atributo especificado dentro de este predicado. Los operadores soportados en los predicados son los siguientes:
    - Operador booleano AND: and
    - Operador booleano OR: or
    - Igual: =
    - No igual: !=
    - Menor que: <
    - Menor o igual que: <=
    - Mayor que: &gt
    - Mayor o igual que: >=
    - Suma: +
    - Resta: -
    - Multiplicación: *
    - División: div
- Ejes. Con los ejes se puede realizar una selección dentro del árbol para seleccionar nodos. Los ejes utilizados en XPath son los siguientes:
    - Child representado por /.
    - Attribute representado por @.
    - Descendant representado por //.
    - Self representado por .
    - Parent representado por ..
    - Ancestor representado por ancestor::

6.3. EJEMPLO DE CREACIÓN DE UNA NUEVA REGLA

Para realizar una nueva regla, primero se debe identificar el patrón sintáctico que sigue la regla, es decir, se debe identificar la estructura que identifica de forma unívoca a la expresión que se quiere encontrar en el código.

Como ejemplo para crear una nueva regla, se hará uso de la regla número 8.12 de la guía de buenas prácticas que marca MISRA para el lenguaje de programación C, y que dice así: "When an array is declared with external linkage, its size shall be stated explicitly or defined implicitly by initialisation ", es decir, cuando se vaya a declarar un array con un enlace externo se debe especificar su tamaño de forma explícita o implícitamente mediante una declaración.

```
int main()
{
     int array[10];
     extern int array1[];
     int array2[]={0,10,15};

    return 0;
}
```

Ilustración 35: Código de apoyo de la regla 18.12 de MISRA C.

En la Ilustración 35 se presenta un fragmento de código en el que la declaración de un array incumple la regla presentada anteriormente, mientras que el resto de declaraciones no la incumplen.

La mejor forma de identificar expresiones es mediante la utilización de una herramienta llamada SSLR C Toolkit. Esta herramienta proporciona un entorno en el que generar árboles sintácticos abstractos e identificar expresiones para definir reglas.

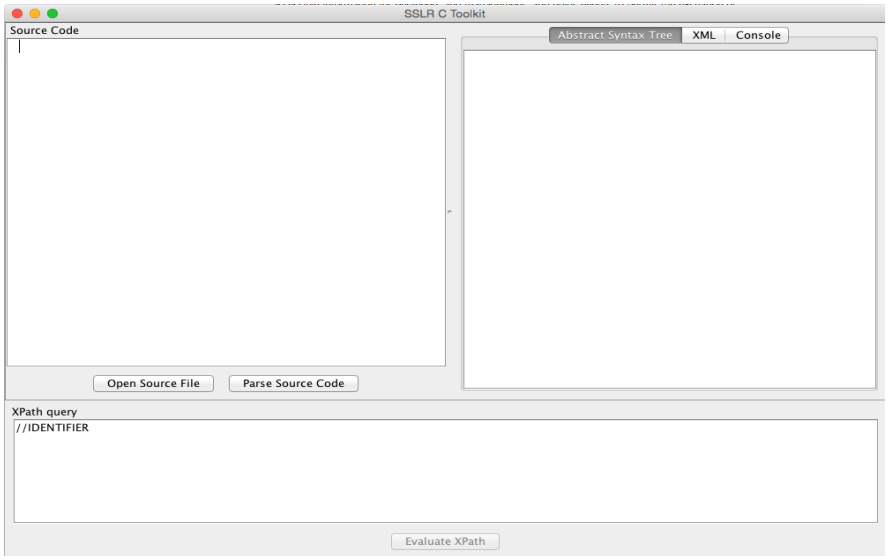


Ilustración 36: Herramienta SSLR C toolkit.

Dentro de la herramienta, presentada en la Ilustración 36, podemos distinguir las siguientes partes:

- Source Code. Es una caja de texto donde se debe introducir el código deseado. La forma de incluir el código en esta sección es manualmente, pegando el código, o abriendo un archivo en el que reside el código.
- Abstract Syntax Tree. Una vez que el código ha sido insertado en su lugar, se procede a la generación del árbol sintáctico mediante la pulsación del botón "Parse Source Code". El árbol sintáctico generado sigue la forma de la imagen 37.
- XPath query: Es el lugar donde se generará la expresión para buscar los trozos de código deseados, en este caso, que incumplen la regla 8.12 del estándar MISRA C.

Ilustración 37: Árbol sintáctico generado para el código de la regla 8.12.

Una vez generado el árbol, se pasa a señalar la declaración que incumple la regla: “extern int array[];”.

Se puede comprobar que en la expresión que marca la creación de un array es la siguiente: //declaration/declarator/directDeclarator. Dicha expresión, identifica las dos primeras instancias incluyendo en la búsqueda una declaración correcta y otra incorrecta. La diferencia que existe entre ambas declaraciones reside en los hijos del nodo directDeclarator, tal y como se muestra en la Ilustración 38. Una declaración correcta de un array de forma explícita requiere un nodo de tipo “expression”, por lo que una declaración incorrecta no tendrá este nodo. Por lo que la expresión que identifica la declaración incorrecta es la siguiente: //declaration/declarator/ directDeclarator [not(expression)].

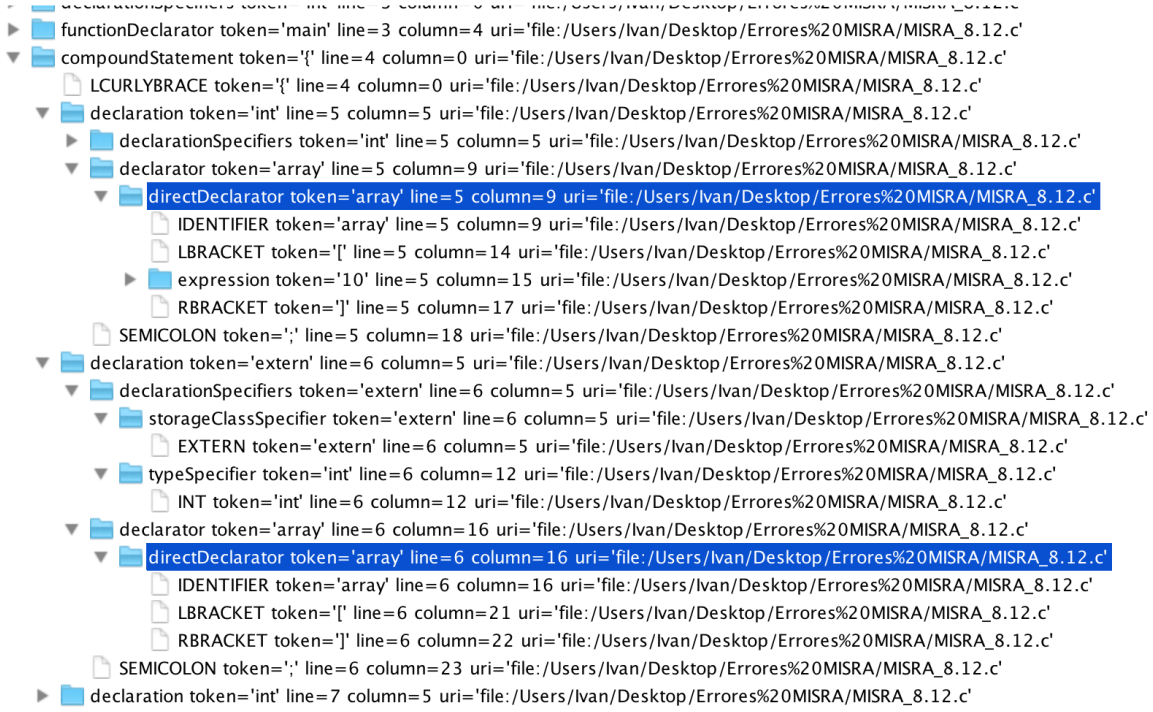


Ilustración 38: Nodos hijos de directDeclarator.

Hay que tener en cuenta que una declaración de una variable tiene la misma estructura que la indicada anteriormente, por lo que se debe añadir una cláusula más para que no se identifiquen declaraciones correctas como incorrectas, por lo tanto, se deben añadir elementos que afinen aún más la búsqueda, estos elementos son los corchetes. Una vez añadidos los corchetes a la expresión, está queda de la siguiente forma: //declaration/declarator/directDeclarator[LBRACKET and not(expression) and RBRACKET]

Para comprobar que la expresión es correcta, se debe hacer uso del apartado XPath query de la aplicación, tal como se muestra en la Ilustración 39.

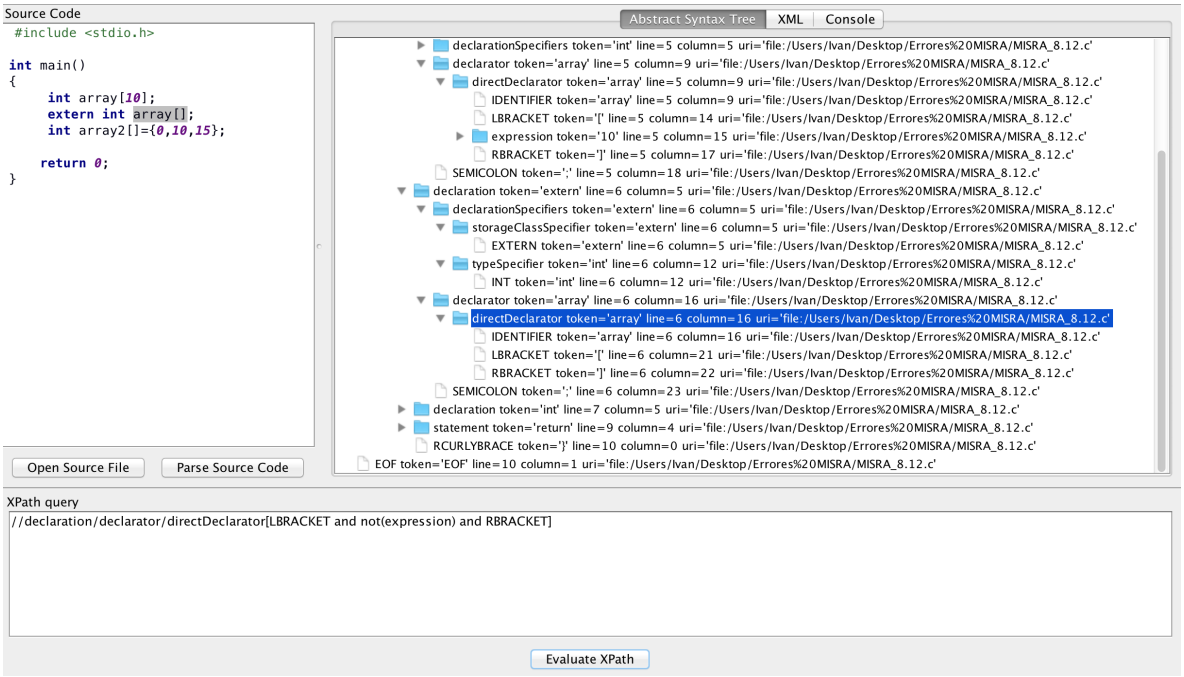


Ilustración 39: Resultado de aplicar la expresión generada.

CAPITULO 7
PRUEBAS

En este capítulo se desarrolla el plan de pruebas y la matriz de trazabilidad para comprobar que el sistema cubre los requisitos propuestos anteriormente.

7.1. DEFINICIÓN DEL PLAN DE PRUEBAS

Todas las pruebas de aceptación del sistema seguirán el modelo presentado en la Tabla 84. Todas las pruebas han sido generadas a partir de los requisitos del sistema, por lo que se establecen por completo las pruebas necesarias para asegurar que el sistema se ha desarrollado correctamente.

ID	
Nombre	
Objetivo	
Descripción	
Resultado obtenido	

Tabla 84: Modelo de tabla para las pruebas del Sistema.

Cada prueba cuenta con los siguientes campos:

- ID: cada una de las pruebas cuenta con un identificador único. Este identificador será utilizado posteriormente para identificar cada prueba en la matriz de trazabilidad de Pruebas con Requisitos.
- Nombre: es el nombre que se le asocia a cada prueba.
- Objetivo: se trata de una breve explicación que muestra que parte del sistema va a cubrir.
- Descripción: se trata de una descripción detallada de la prueba.
- Resultado obtenido: se trata de una breve descripción que se obtiene al realizar la prueba en el sistema.

A continuación se presentan todas las pruebas del sistema:

PA-01	
Nombre	Registro de usuario.
Objetivo	Comprobar que el sistema permite a los usuarios registrarse.
Descripción	En la pantalla de registro del sistema el usuario inserta sus datos personales para crear una cuenta y presiona el botón de registro.
Resultado obtenido	El registro de usuario se realiza correctamente. Además, tras el registro se realiza el acceso al sistema con los datos del usuario.

Tabla 85: Prueba de aceptación del sistema 01.

PA-02	
Nombre	Acceso de usuarios.
Objetivo	Comprobar que el sistema permite acceder a los usuarios registrados.
Descripción	En la pantalla de login, el usuario registrado inserta sus datos personales y al presionar sobre el botón de registro accede al sistema.
Resultado obtenido	El acceso de los usuarios al sistema es correcto.

Tabla 86: Prueba de aceptación del sistema 02.

PA-03	
Nombre	Finalización de sesión.
Objetivo	Comprobar que el sistema elimina correctamente los datos del usuario de la sesión.
Descripción	Dentro del sistema, el usuario selecciona cerrar la sesión e intenta acceder a cualquier parte del sistema mediante la barra del explorador.
Resultado obtenido	El usuario realiza el cierre de sesión correctamente. Además, cualquier acceso al sistema sin un usuario registrado se redirige a la ventana de login.

Tabla 87: Prueba de aceptación del sistema 03.

PA-04	
Nombre	Mostrar formulario para la subida de proyectos.
Objetivo	Comprobar que el sistema muestra al usuario el formulario para subir los proyectos.
Descripción	Dentro del sistema, el usuario selecciona subir un proyecto y se muestra el formulario que es necesario rellenar para realizar la subida.
Resultado obtenido	El formulario para la subida de proyectos se muestra correctamente.

Tabla 88: Prueba de aceptación del sistema 04.

PA-05	
Nombre	Mostrar formulario para la subida de versiones.
Objetivo	Comprobar que el sistema muestra al usuario el formulario para subir nuevas versiones.
Descripción	Dentro del sistema, el usuario selecciona subir una versión y se muestra el formulario que es necesario rellenar para realizar la subida.
Resultado obtenido	El formulario para realizar la subida de versiones se muestra correctamente.

Tabla 89: Prueba de aceptación del sistema 05.

PA-06	
Nombre	Mostrar formulario para la creación de informes.
Objetivo	Comprobar que el sistema muestra al usuario el formulario para realizar un informe.
Descripción	Dentro del sistema, el usuario selecciona crear un informe y se muestra el formulario que es necesario realizar el informe.
Resultado obtenido	El formulario para analizar un proyecto y crear informes se muestra correctamente.

Tabla 90: Prueba de aceptación del sistema 06.

PA-07	
Nombre	Mostrar lista con los informes generados.
Objetivo	Comprobar que el sistema muestra al usuario los informes generados.
Descripción	Dentro del sistema, el usuario selecciona visualizar un informe, selecciona el proyecto deseado y se muestran todos los informes generados para ese proyecto.
Resultado obtenido	Se muestra correctamente la lista de informes generados con opciones de visualizar y descargar.

Tabla 91: Prueba de aceptación del sistema 07.

PA-08	
Nombre	Mostrar proyectos para visualizar su evolución.
Objetivo	Comprobar que el sistema muestra al usuario los proyectos analizados para visualizar su evolución.
Descripción	Dentro del sistema, el usuario selecciona mostrar evolución y se presenta una lista con todos aquellos proyectos analizados.
Resultado obtenido	La lista con los proyectos analizados se muestra correctamente, y, no aparecen proyectos que no han sido analizados.

Tabla 92: Prueba de aceptación del sistema 08.

PA-09	
Nombre	Mostrar proyectos para realizar comparaciones.
Objetivo	Comprobar que el sistema muestra al usuario los proyectos analizados para realizar comparaciones.
Descripción	Dentro del sistema, el usuario selecciona comparar proyectos y se presenta una lista con todos aquellos proyectos analizados.
Resultado obtenido	La lista con los proyectos analizados se muestra correctamente, y, no aparecen proyectos que no han sido analizados.

Tabla 93: Prueba de aceptación del sistema 09.

PA-10	
Nombre	Completar formulario para subir un proyecto.
Objetivo	Comprobar que el sistema permite al usuario seleccionar la versión del proyecto y, el archivo comprimido en formato zip con el código fuente.
Descripción	Dentro del sistema, el usuario, después de seleccionar subir un proyecto, puede seleccionar la versión del proyecto y el archivo comprimido que contiene su código fuente.
Resultado obtenido	El usuario puede seleccionar la versión y los ficheros comprimidos correctamente.

Tabla 94: Prueba de aceptación del sistema 10.

PA-11	
Nombre	Realizar subida de un proyecto.
Objetivo	Comprobar que el sistema realiza subidas de proyectos.
Descripción	Dentro del sistema, el usuario, después de seleccionar subir un proyecto, rellenar el formulario y seleccionar subir proyecto, recibe un mensaje con el resultado de la operación.
Resultado obtenido	Si los datos se completan correctamente y el proyecto está comprimido en un archivo .zip, la subida se realiza correctamente, por lo que se muestra un mensaje al usuario. En caso contrario, se muestra un mensaje notificando el error.

Tabla 95: Prueba de aceptación del sistema 11.

PA-12	
Nombre	Reestablecer el proceso de subida de un proyecto.
Objetivo	Comprobar que el sistema reestablece el formulario de subida de un proyecto.
Descripción	Dentro del sistema, el usuario, después de seleccionar subir un proyecto puede reestablecer el formulario siempre que desee.
Resultado obtenido	El botón de reset funciona correctamente y reinicia todos los campos del formulario.

Tabla 96: Prueba de aceptación del sistema 12.

PA-13	
Nombre	Completar formulario para subir una versión.
Objetivo	Comprobar que el sistema permite al usuario seleccionar un proyecto, la nueva versión y el archivo comprimido en formato zip con el código fuente.
Descripción	Dentro del sistema, el usuario, después de seleccionar subir versión, puede seleccionar el proyecto, la versión y el archivo comprimido que contiene su código fuente.
Resultado obtenido	Todos los campos se pueden rellenar correctamente. Además, el selector de los proyectos muestra todos los proyectos que el usuario ha subido anteriormente.

Tabla 97: Prueba de aceptación del sistema 13.

PA-14	
Nombre	Realizar subida de una versión.
Objetivo	Comprobar que el sistema realiza la subida de una versión.
Descripción	Dentro del sistema, el usuario, después de seleccionar subir una versión, rellenar el formulario y seleccionar subir proyecto recibe un mensaje con el resultado de la operación.
Resultado obtenido	Si los datos se completan correctamente y el proyecto está comprimido en un archivo .zip, la subida se realiza correctamente, por lo que se muestra un mensaje al usuario. En caso contrario, se muestra un mensaje notificando el error.

Tabla 98: Prueba de aceptación del sistema 14.

PA-15	
Nombre	Reestablecer el proceso de subida de una versión.
Objetivo	Comprobar que el sistema reestablece el formulario de subida de una versión.
Descripción	Dentro del sistema, el usuario, después de seleccionar subir una versión puede reestablecer el formulario siempre que desee.
Resultado obtenido	El botón de reset funciona correctamente y reinicia todos los campos del formulario.

Tabla 99: Prueba de aceptación del sistema 15.

PA-16	
Nombre	Mostrar versiones subidas de un proyecto.
Objetivo	Comprobar que el sistema muestra las versiones subidas de un proyecto.
Descripción	Dentro del sistema, el usuario, después de seleccionar crear un informe, selecciona un proyecto y se cargan sus versiones subidas.
Resultado obtenido	Se cargan correctamente las versiones subidas del proyecto.

Tabla 100: Prueba de aceptación del sistema 16.

PA-17	
Nombre	Mostrar perfiles de un lenguaje.
Objetivo	Comprobar que el sistema muestra los perfiles de un lenguaje de programación.
Descripción	Dentro del sistema, el usuario, después de seleccionar crear un informe, selecciona un lenguaje de programación y se cargan sus perfiles de reglas.
Resultado obtenido	La carga de perfiles se realiza correctamente.

Tabla 101: Prueba de aceptación del sistema 17.

PA-18	
Nombre	Completar formulario para crear un informe.
Objetivo	Comprobar que el sistema permite al usuario completar el formulario para crear un informe.
Descripción	Dentro del sistema, el usuario, después de seleccionar crear informe, puede seleccionar el proyecto, la versión, el lenguaje, el perfil y los campos opcionales (Multiproyecto y añadir perfil).
Resultado obtenido	La selección de todos los elementos del formulario de creación de informes funciona correctamente.

Tabla 102: Prueba de aceptación del sistema 18.

PA-19	
Nombre	Crear informe de un proyecto.
Objetivo	Comprobar que el sistema permite crear informes.
Descripción	Dentro del sistema, el usuario, después de seleccionar crear informe y rellenar el formulario presiona crear informe. Tras realizar el análisis se muestra el informe generado.
Resultado obtenido	Tras la creación del informe,éste se muestra correctamente. En caso de que se produzca un error durante el análisis y la creación se notifica al usuario.

Tabla 103: Prueba de aceptación del sistema 19.

PA-20	
Nombre	Mostrar evolución de un proyecto.
Objetivo	Comprobar que el sistema muestra al usuario la evolución de un proyecto.
Descripción	Dentro del sistema, el usuario selecciona mostrar evolución, selecciona un proyectos analizado y se muestra la evolución de un proyecto.
Resultado obtenido	Los datos de evolución se muestran correctamente. La evolución de errores se muestra en una gráfica de barras y el resto de valores en una tabla com medias y desviaciones tipicas.

Tabla 104: Prueba de aceptación del sistema 20.

PA-21	
Nombre	Completar formulario para comparar informe.
Objetivo	Comprobar que el sistema permite al usuario seleccionar varios proyectos analizados para realizar una comparación.
Descripción	Dentro del sistema, después de seleccionar comparar proyectos, se muestra una lista con los proyectos analizados. Para esa lista, se puede seleccionar cualquier conjunto de los proyectos mostrados para hacer la comparación.
Resultado obtenido	La lista y la selección de uno o varios proyectos se realiza correctamente.

Tabla 105: Prueba de aceptación del sistema 21.

PA-22	
Nombre	Mostrar datos de la comparación
Objetivo	Comprobar que el sistema muestra los datos de la comparación de los proyectos deseados.
Descripción	Dentro del sistema, el usuario, después de seleccionar comparar proyectos y de haber seleccionado al menos uno, se muestran todos los datos relativos a la comparación.
Resultado obtenido	Los datos de la comparación se muestran correctamente, así como los cálculos que se realizan para la media y la desviación típica.

Tabla 106: Prueba de aceptación del sistema 22.

PA-23	
Nombre	Despliegue del Sistema.
Objetivo	Comprobar que el sistema es desplegado en un Sistema Cloud.
Descripción	Se facilitará una captura del estado de la máquina en el Sistema Cloud.
Resultado obtenido	Como se puede comprobar en la Ilustración 23, el Sistema ha sido desplegado en un entorno cloud.

Tabla 107: Prueba de aceptación del sistema 23.

PA-24	
Nombre	Replicación automática de datos.
Objetivo	Comprobar que el sistema replica automáticamente todos los datos del Sistema.
Descripción	Se facilitará una captura del estado de la máquina en el Sistema Cloud.
Resultado obtenido	Como el Sistema Cloud con OpenStack tiene por debajo un sistema de almacenamiento distribuido Cepth, el cual asegura un mínimo de tres copias por cada objeto (para este sistema, máquina virtual y volumen de datos), la replicación de datos queda cubierta.

Tabla 108: Prueba de aceptación del sistema 24.

PA-25	
Nombre	Configuración de la instancia.
Objetivo	Comprobar que el sistema cuenta con 6 Gigabytes de RAM y 40 Gigasbytes de almacenamiento.
Descripción	Se facilitará una captura del estado de la máquina en el Sistema Cloud.
Resultado obtenido	Como el Sistema Cloud con OpenStack tiene por debajo un sistema de almacenamiento distribuido cepth, el cual asegura un mínimo de tres copias por cada objeto (para este sistema, máquina virtual y volumen de datos), la replicación de datos queda cubierta.

Tabla 109: Prueba de aceptación del sistema 25.

PA-26	
Nombre	Tamaño del volumen de datos.
Objetivo	Comprobar que el volumen de datos asociado a la instancia cuenta con el almacenamiento especificado.
Descripción	Se facilitará una captura del volumen de datos para verificar que cuenta con 40 Gigabytes de almacenamiento.
Resultado obtenido	Como el Sistema Cloud con OpenStack tiene por debajo un sistema de almacenamiento distribuido cepth, el cual asegura un mínimo de tres copias por cada objeto (para este sistema, máquina virtual y volumen de datos), la replicación de datos queda cubierta.

Tabla 110: Prueba de aceptación del sistema 26.

PA-27	
Nombre	Versión instalada de Java.
Objetivo	Comprobar que la versión instalada de Java es compatible con SonarQube.
Descripción	En la documentación interna de SonarQube se establece la versión mínima de Java para que funcione correctamente.
Resultado obtenido	Dado que el Sistema analiza y crea los informes mediante SonarQube, la versión de Java es correcta. Además, en la Ilustración 33 se muestra la versión de Java instalada en el Sistema.

Tabla 111: Prueba de aceptación del sistema 27.

PA-28	
Nombre	Base de datos.
Objetivo	Demostrar que el gestor de bases de datos instalado es MySQL.
Descripción	Se facilitará una captura de pantalla en la que se muestra como las tablas se encuentran almacenadas en MySQL.
Resultado obtenido	En la Ilustración 35 se puede comprobar como dentro del gestor MySQL se encuentran todas las tablas que presenta Sonar, las cuales se utilizan también en este Sistema.

Tabla 112: Prueba de aceptación del sistema 28.

PA-29	
Nombre	Comprobar login único de usuario.
Objetivo	Demostrar que el login del usuario es único.
Descripción	Una vez registrado en el sistema, se cierra sesión y se vuelve a registrar con el mismo login.
Resultado obtenido	El Sistema avisa de que ese login ya está en uso, por lo que se aborta la creación del usuario.

Tabla 113: Prueba de aceptación del sistema 29.

PA-30	
Nombre	Archivo que contiene el código fuente del proyecto.
Objetivo	Demostrar que el Sistema sólo acepta ficheros comprimidos .zip.
Descripción	El usuario intentará subir el mismo código comprimido en varios formatos de compresión.
Resultado obtenido	El Sistema solo permite la subida de ficheros .zip, y aborta la subida para el resto de tipos.

Tabla 114: Prueba de aceptación del sistema 30.

PA-31	
Nombre	Exploradores web.
Objetivo	Demostrar que el Sistema funciona correctamente en todos los exploradores web.
Descripción	El usuario realizará las mismas operaciones en distintos navegadores.
Resultado obtenido	En todos los navegadores el Sistema tiene el mismo comportamiento.

Tabla 115: Prueba de aceptación del sistema 31.

PA-32	
Nombre	Gráficos de barras en las comparaciones.
Objetivo	Comprobar que la evolución de los errores de un proyecto se muestra mediante un gráfico de barras.
Descripción	El usuario selecciona un proyecto para ver su evolución y comprueba que el gráfico de errores es de barras.
Resultado obtenido	El gráfico concuerda con lo descrito en la prueba.

Tabla 116: Prueba de aceptación del sistema 32.

PA-33	
Nombre	Comprobación de las tablas.
Objetivo	Comprobar que la evolución de un proyecto se muestra en tablas con medias y desviaciones tipicas.
Descripción	El usuario selecciona un proyecto para ver su evolución y comprueba que los datos de complejidad se muestran en una tabla.
Resultado obtenido	La tabla representa los valores correctamente. Además los cálculos de la media y desviaciones son correctos

Tabla 117: Prueba de aceptación del sistema 33.

PA-34	
Nombre	Barra lateral del Sistema.
Objetivo	Comprobar que el Sistema muestra siempre al usuario una barra con las opciones principales.
Descripción	Durante la navegación del usuario por el sistema, siempre se muestra el logotipo del grupo de investigación y las opciones principales que puede realizar el usuario.
Resultado obtenido	El logotipo y las opciones siempre están visibles.

Tabla 118: Prueba de aceptación del sistema 34.

PA-35	
Nombre	Conexiones seguras.
Objetivo	Comprobar que el Sistema realiza conexiones seguras.
Descripción	El usuario accede al sistema mediante el protocolo de transferencia seguro HTTPS.
Resultado obtenido	El sistema permite realizar conexiones seguras, además muestra los datos del certificado digital al usuario.

Tabla 119: Prueba de aceptación del sistema 35.

PA-36	
Nombre	Almacenamiento seguro de contraseñas.
Objetivo	Comprobar que el Sistema almacena las contraseñas de forma segura.
Descripción	Todas las páginas PHP encargadas de realizar registros y accesos, no utilizan las contraseñas en claro sino funciones resumen.
Resultado obtenido	Durante la implementación del Sistema, se utiliza el mecanismo de función resumen SHA1, para evitar que las contraseñas se utilicen en claro en el Sistema.

Tabla 120: Prueba de aceptación del sistema 36.

PA-37	
Nombre	Atributos cifrados al utilizar el paso de parámetro mediante Request.
Objetivo	Comprobar que el sistema al utilizar el paso de parámetros mediante Request, todos los datos son cifrados y no viajan en claro.
Descripción	El usuario al realizar acceso a la visualización de informes, el parámetro archive va cifrado.
Resultado obtenido	Todos los parámetros de la Request van cifrados.

Tabla 121: Prueba de aceptación del sistema 37.

En las siguientes ilustraciones se muestra parte de los resultados de las pruebas presentadas anteriormente:

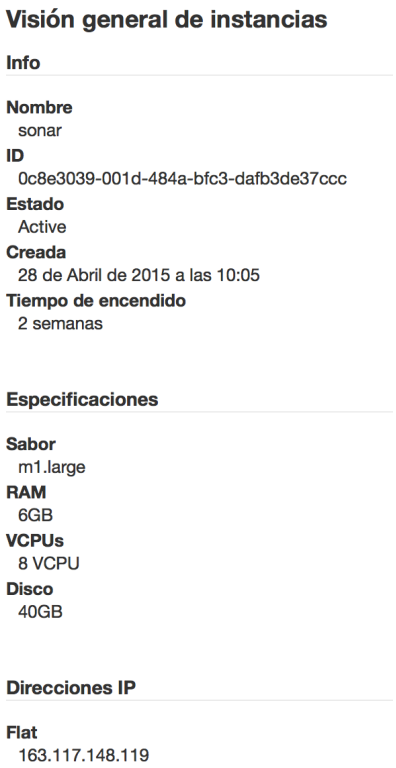


Ilustración 40: Características de la instancia sobre la que se desarrolla el Sistema.

Como se puede observar en la Ilustración número 40, la instancia virtual creada cuenta con 8 procesadores virtuales, 6 Gigabytes de memoria RAM y 40 Gigabytes de disco duro. Se puede observar el día de creación de la instancia. Es importante destacar que este día no concuerda con el período que aparece en la planificación, ya que se produjo un error en esa semana de abril, y como consecuencia la otra instancia quedó inservible, pero, los datos no sufrieron ningún tipo de daño porque el volumen de datos es independiente de la instancia.

```
root@sonar:/home/sonar# java -version
java version "1.7.0_75"
OpenJDK Runtime Environment (IcedTea 2.5.4) (7u75-2.5.4-1~trusty1)
OpenJDK 64-Bit Server VM (build 24.75-b04, mixed mode)
root@sonar:/home/sonar#
```

Ilustración 41: Versión de Java instalada.

En la Ilustración 41, se puede comprobar que la versión instalada de java es superior a la mínima que establece SonarQube, por lo que la instancia virtual cumple las especificaciones dictadas por los desarrolladores.

```
| datadir                                 | /var/lib/mysql/
| date_format                             | %Y-%m-%d
| datetime_format                         | %Y-%m-%d %H:%i:%s
| default_storage_engine                  | InnoDB
| default_week_format                     | 0
| delay_key_write                         | ON
```

Ilustración 42: Motor configurado por defecto en MySQL.

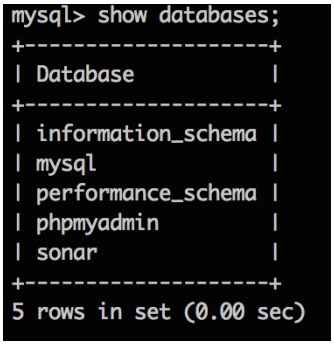


Ilustración 43: Bases de datos creadas en MySQL.

La Ilustración 42 e Ilustración 43 muestran las configuraciones básicas de MySQL, como el motor de almacenamiento predefinido, InnoBD, y como existe la base de datos Sonar, la cual utiliza el SonarQube para almacenar el resultados de los análisis, usuarios y los perfiles de reglas.

7.2. MATRIZ DE TRAZABILIDAD

En este apartado se muestra una tabla que representa la relación existente entre las pruebas de aceptación realizadas y los requisitos descriptor en la fase de análisis. Como se puede comprobar en la Tabla 122.1 y Tabla 122.2, el sistema queda completamente cubierto por las pruebas de aceptación.

RF/PA	PA-01	PA-02	PA-03	PA-04	PA-05	PA-06	PA-07	PA-08	PA-09	PA-10	PA-11	PA-12	PA-13	PA-14	PA-15	PA-16	PA-17	PA-18	PA-19
RF-01	X																		
RF-02		X																	
RF-03			X																
RF-04				X															
RF-05					X														
RF-06						X													
RF-07							X												
RF-08								X											
RF-09									X										
RF-10										X									
RF-11										X									
RF-12											X								
RF-13											X								
RF-14												X							
RF-15													X						
RF-16													X						
RF-17													X						
RF-18														X					
RF-19														X					
RF-20															X				
RF-21																X		X	
RF-22																		X	
RF-23																		X	
RF-24																		X	
RF-25																	X	X	
RF-26																		X	
RF-27																			X
RF-28																			X

Tabla 122.1: Matriz de trazabilidad de requisitos funcionales y pruebas de aceptación.

RF/PA	PA-20	PA-21	PA-22	PA-23	PA-24	PA-25	PA-26	PA-27	PA-28	PA-29	PA-30	PA-31	PA-32	PA-33	PA-34	PA-35	PA-36	PA-37
RF-29	X																	
RF-30		X																
RF-31			X															
RR-01				X														
RR-02					X													
RR-03						X												
RR-04						X	X											
RR-05								X										
RR-06									X									
RR-07									X									
RR-08									X									
RR-09										X								
RR-10											X							
RR-11												X						
RR-12													X					
RR-13														X				
RI-01															X			
RI-02															X			
RS-01																X		
RS-02																	X	
RS-03																		X

Tabla 122.2: Matriz de trazabilidad de requisitos funcionales y pruebas de aceptación.

CAPITULO 8
EVALUACIÓN DEL SISTEMA

En este capítulo se desarrolla la evaluación llevada a cabo al sistema con la ayuda del grupo de investigación de Arquitectura de Computadores Comunicaciones y Sistemas, ARCOS dentro la Universidad Carlos III de Madrid.

Para realizar la evaluación se han elegido las siguientes prácticas realizadas por alumnos de la universidad Carlos III de Madrid:

- Asignatura Sistemas Operativos, 2º curso, laboratorio número 3, grupo 89.
- Asignatura Diseño de Sistemas Operativos, 3er curso, laboratorio número 1, grupo 89.
- Asignatura Sistemas Distribuidos, 3er curso, laboratorio 2, grupo 89.

Primero, cada uno de los laboratorios serán analizados independientemente, con el fin de obtener una relación entre la duración del análisis invertido con la complejidad y el tamaño del código.

Después, se compararán los valores medios de cada laboratorio con el fin de obtener una idea de la calidad del código y los tiempos medios de análisis.

8.1. LABORATORIO NÚMERO 3 DE SISTEMAS OPERATIVOS

Este laboratorio se compone de un total de 19 prácticas. Los valores asociados al tiempo, complejidad y líneas de código se pueden observar en la Tabla 123. El tiempo medio de análisis, por práctica, para este laboratorio es de 11,32 segundos. Y el tiempo total invertido en analizar todo el laboratorio es de 215,089 segundos.

Es muy importante destacar las prácticas 1, 11 y 15 que sólo poseen valores en el tiempo y no en el resto de métricas. Estos valores nulos se obtiene al producirse un error en el análisis realizado por SonarQube, es decir, las tres prácticas poseen errores de compilación, por lo que SonarQube no es capaz de analizar la calidad del código.

Proyecto	Tiempo	Complejidad	Líneas
1	6,19		
2	12,54	81	836
3	12,78	77	721
4	12,14	65	697
5	11,63	33	448
6	13,07	43	607
7	12,17	42	533
8	12,61	37	423
9	11,67	58	530
10	11,82	49	562
11	6,28		
12	12,45	70	771
13	12,93	54	496
14	12,56	65	626
15	5,99		
16	12,42	47	584
17	11,56	37	423
18	12,01	56	565
19	12,29	81	628

Tabla 123: Resultados obtenidos en el tercer laboratorio de la asignatura de Sistemas Operativos.

8.1.1. INTERPRETACIÓN DE LOS DATOS OBTENIDOS

En este apartado se interpretan los datos obtenidos en los análisis de las prácticas. Para ello, se comparará el valor obtenido en el tiempo con el resto de valores (complejidad y líneas de código)

Como se puede comprobar en la Ilustración 44, la duración del análisis de cada práctica es independiente de la complejidad que presenta el código. Además, tampoco depende directamente del tamaño de cada una de las prácticas, ya que tampoco se aprecia una relación evidente entre ambos valores.

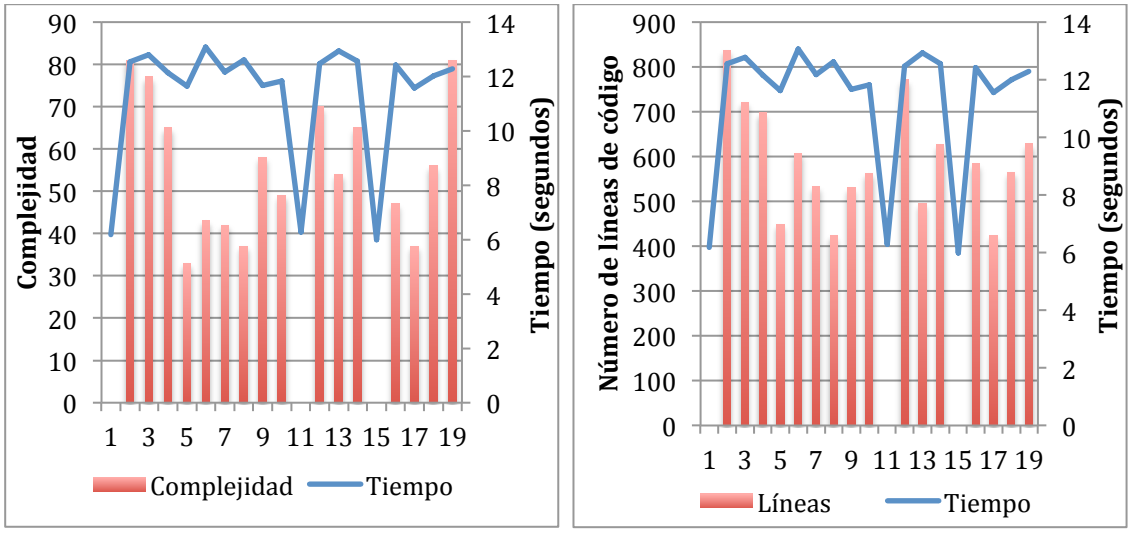


Ilustración 44: Gráficas con la representación del tiempo con el resto de valores para cada práctica.

8.2. Laboratorio número 2 de Sistemas Distribuidos

En este apartado se analiza el código entregado por los alumnos del grupo 89 de la asignatura de Sistemas Distribuidos. Este laboratorio se compone de un total de 12 grupos de prácticas. Los datos relevantes para su evaluación se encuentran expuestos en la Tabla 124.

Proyecto	Tiempo	Complejidad	Líneas
1	19,30	628,00	4200,00
2	12,15	53,00	422,00
3	6,16		
4	11,81	70,00	726,00
5	12,78	67,00	723,00
6	11,50	62,00	642,00
7	14,13	84,00	570,00
8	3,93		
9	6,36		
10	12,48	98,00	714,00
11	12,22	55,00	497,00
12	13,54	51,00	540,00

Tabla 124: Resultados obtenidos en el segundo laboratorio de la asignatura de Sistemas Distribuidos.

8.2.1. Interpretación de los datos obtenidos

Como en el caso anterior, existen varias prácticas que contienen fallos de compilación, por lo que el análisis realizado por Sonar falla. Como se puede comprobar en la Ilustración 45, no se encuentra una relación muy evidente entre el tiempo empleado en el análisis con la complejidad y el tamaño de cada práctica.

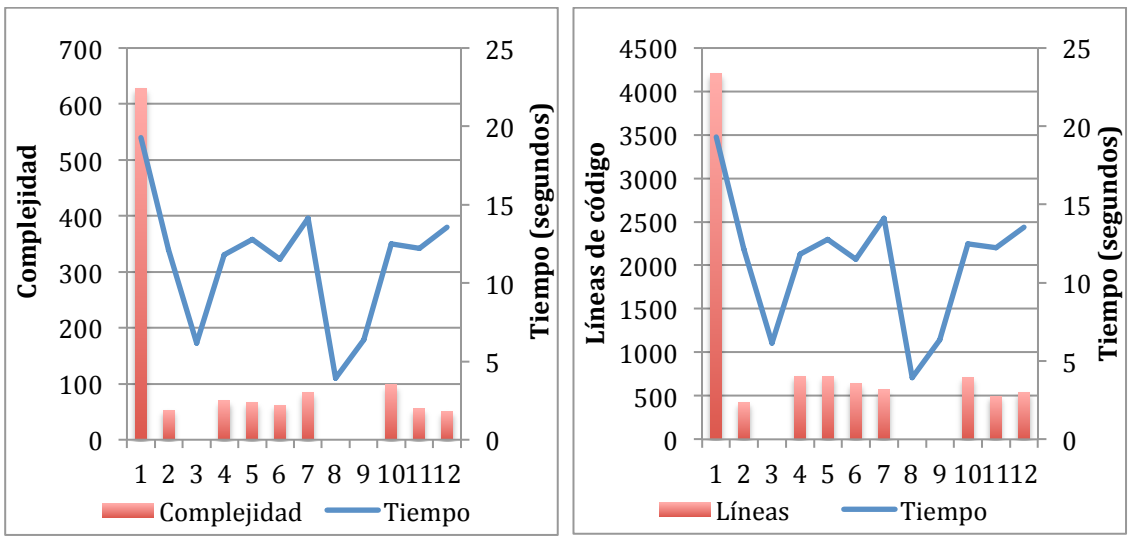


Ilustración 45: Gráficas con la representación del tiempo con el resto de valores para cada práctica.

8.3. Laboratorio número 1 de Diseño de Sistemas Operativos

El laboratorio de Diseño de Sistemas Operativos está compuesto por un total de 11 prácticas. De todas estas prácticas hay que destacar que solamente una no compila y que otra tiene unos valores muy elevados en tiempo complejidad y errores, tal y como se puede comprobar en la Tabla 125.

Proyecto	Tiempo	Complejidad	Líneas
1	14,74	166,00	1044,00
2	36,54	4751,00	24554,00
3	14,66	160,00	1092,00
4	14,21	161,00	933,00
5	14,13	179,00	1020,00
6	12,99	155,00	1005,00
7	13,37	184,00	1006,00
8	12,71	110,00	587,00
9	6,00		
10	13,28	161,00	933,00
11	13,67	110,00	587,00

Tabla 125: Resultados obtenidos en el primer laboratorio de la asignatura de Diseño de Sistemas Operativos.

8.3.1. Interpretación de los datos obtenidos

A diferencia de los casos anteriores, la relación existente entre el tiempo y el tamaño de la práctica es mas evidente, sin embargo la duración es totalmente independiente de la complejidad del código analizado, tal y como se puede comprobar en la Ilustración 46.

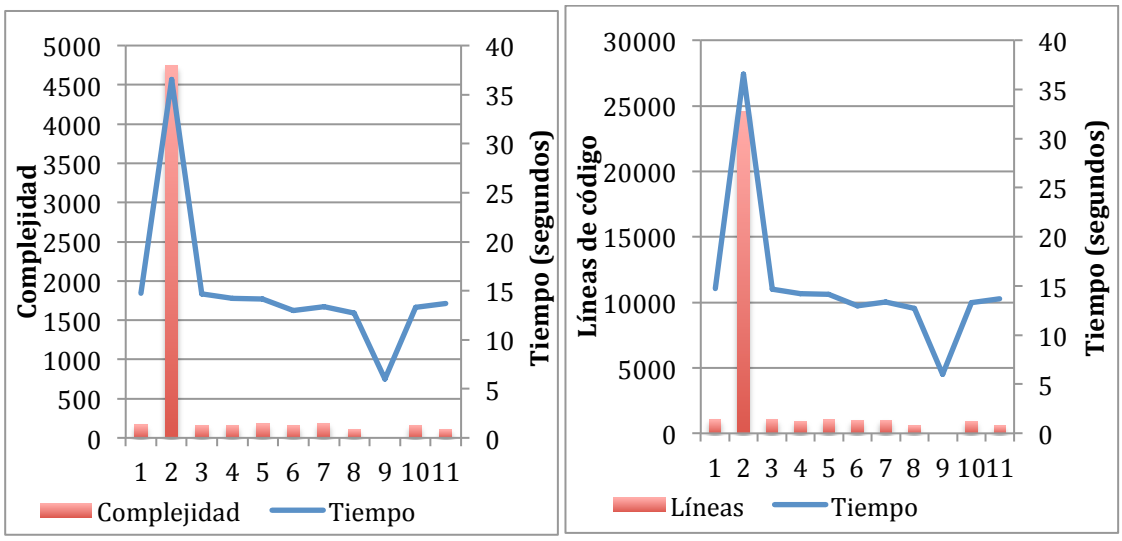


Ilustración 46: Gráficas con la representación del tiempo con el resto de valores para cada práctica.

8.4 ANÁLISIS GLOBAL DE LOS DATOS RECOGIDOS

En este apartado se proporciona un análisis de los valores medios y desviaciones calculadas para todos los laboratorios analizados. Como se puede comprobar en la Tabla 126, el tiempo medio de análisis para un código pequeño, alrededor de 1.000 líneas de código o menos, es de 11,32 segundos. Su alto nivel de desviación viene determinado por las prácticas que no compilan, las cuales llevan asociado un tiempo muy bajo.

En cuanto a la complejidad, se puede comprobar como la media es cada vez mayor al analizar prácticas de cursos posteriores. Este hecho se produce porque cada año las prácticas son más complicadas y exigen una mayor abstracción por parte de los alumnos, de ahí que los valores de las desviaciones también aumenten.

Por último, el valor de la media en las líneas de código y las desviaciones asociadas indican que existe una gran dispersión en los datos tomados, por lo que para un laboratorio con una funcionalidad concreta, la diferencia en el tamaño del código es muy elevada.

Laboratorio	Tiempo		Complexity		Lines of Code	
	Media	Desviación	Media	Desviación	Media	Desviación
SSOO Lab1	11,32	2,34	55,94	15,92	590,63	120,55
SSDD Lab2	11,36	4,13	129,78	187,46	1003,78	1203,39
DSSOO Lab1	15,12	7,50	613,70	1453,92	3276,10	7478,45

Tabla 126: Medias y desviaciones para los datos de los tres laboratorios analizados.

CAPITULO 9
CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se desarrollan todas las conclusiones extraídas durante la realización de este Trabajo Final de Grado. Además, se mostrarán todas las líneas futuras que mejorarían el sistema.

9.1. CONCLUSIONES

Teniendo en cuenta los objetivos marcados al comienzo de este documento, los cuales son:

- Gestión de usuarios y estructuración de tal forma que los datos de un usuario sean privados y no puedan ser consultados por ningún otro.
- Dotar a la infraestructura de un subsistema que se encargue de gestionar la subida de proyectos y las sucesivas versiones de proyectos.
- Facilitar a los usuarios el análisis del código subido en la plataforma.
- Facilitar a los usuarios el seguimiento de la calidad de los proyectos analizados.
- Desplegar la solución en una plataforma flexible.

Tras haber finalizado el desarrollo del sistema, se puede afirmar que los objetivos marcados al comienzo del documento se han cumplido totalmente y satisfactoriamente. Además, este trabajo ha permitido aplicar conocimientos que se adquieren a lo largo de la etapa de estudios en la Universidad.

Por otra parte, la utilización de un *sistema cloud*, actualmente en auge, ha sido un gran valor añadido, ya que es una tecnología que se escapa de los contenidos del Grado. Además, el lenguaje de programación PHP, el cual es muy utilizado en el mundo empresarial, ha permitido comprobar la curva de aprendizaje que se necesita para escribir código operativo con un nuevo lenguaje.

Por último, cabe destacar que el sistema desarrollado necesitará en un futuro mejoras que permitan que siga funcionando correctamente. Además de ir actualizando cada uno de los componentes para dotar al sistema de nuevas funcionalidades y de seguridad adicional.

9.2. Líneas futuras

En esta sección se enumeran y comentan diversas líneas para mejorar el sistema presentado en este Trabajo Final de Grado.

9.2.1. Paralelizar los accesos a bases de datos

Por defecto, SonarQube con el gestor de bases de datos de MySQL e InnoDB como mecanismo de almacenamiento no permite accesos simultáneos a los datos. Por este motivo, al intentar paralelizar el análisis, en ocasiones se producen errores, por lo que se aborta el proceso. El componente SonarQube es de código abierto permitiendo así modificar su código para dotar de cerrojos a la parte que accede a los datos.

9.2.2. Incluir máquinas para analizar en Windows

En el sistema desarrollado se permite analizar proyectos programados en C#, bajo un escueto perfil de reglas. Adicionalmente, SonarQube permite añadir reglas del repositorio que ofrece FxCop. FxCop es una herramienta para analizar código .NET, pero su principal desventaja es que solo se puede instalar bajo el sistema Operativo Windows. Por todo esto, una posible mejora sería la de crear un *pool* de máquinas con sistemas SonarQube que corran bajo distintos sistemas operativos para ofrecer un análisis más preciso a cada lenguaje de programación, ya que se podrán instalar un mayor número de herramientas para aumentar los perfiles.

9.2.3. Creación de un perfil personalizado

La aplicación SonarQube permite incluir nuevas reglas de análisis estático. Una posible mejora consistiría en implementar un perfil con las reglas de buenas prácticas que establece MISRA (para C y C++), ISO/IEC 25000 o perfiles personalizados para que los alumnos verifiquen la calidad del código de sus prácticas.

CAPITULO 10
PLANIFICACIÓN Y PRESUPUESTO

En este capítulo se presenta una descripción de la planificación inicial marcada para el TFG y la planificación real seguida. Además, se mostrará el presupuesto generado durante la realización del TFG y el entorno socio-económico en el que se sitúa este proyecto.

10.1. PLANIFICACIÓN INICIAL

En este apartado se muestran las diversas etapas que ha seguido el TFG desde la fecha de inicio, 15 de diciembre de 2014, hasta la fecha de finalización estimada, 12 de junio de 2015.

Actividad	Fecha de Inicio	Ficha de Fin	Duración(Días)	Horas dedicadas
Plan operativo	15-12-2014	22-12-2014	7	40
Análisis	22-12-2014	8-1-2015	18	44
Diseño	9-1-2015	30-1-2015	24	64
Implementación	31-1-2015	1-4-2015	70	143
Pruebas	2-4-2015	27-4-2015	68	143
Evaluación	28-4-2015	18-5-2015	21	56
Documentación	19-5-2015	5-6-2015	146	390
Revisión	6-6-2015	12-6-2015	12	64

Tabla 127: Planificación inicial.

Para ofrecer una planificación más visual, se ofrece un gráfico Gantt con los datos de la planificación inicial, Tabla 128.1 y Tabla 128.2.

			Diciembre			Enero			
Actividad	Inicio	Fin	15-21	22-28	29-04	05-11	12-18	19-25	26-0
Plan operativo	15-12-2014	22-12-2014							
Análisis	22-12-2014	8-1-2015							
Diseño	9-1-2015	30-1-2015							
Implementación	2-2-2015	12-4-2015							
Pruebas	2-2-2015	27-4-2015							
Evaluación	28-4-2015	18-5-2015							
Documentación	2-2-2015	31-5-2015							
Revisión	1-6-2015	12-6-2015							

Tabla 128.1: Gráfico Gantt de la planificación inicial.

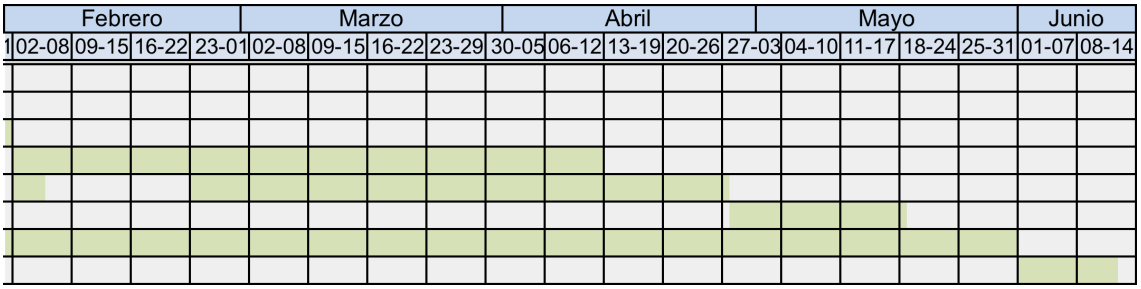


Tabla 128.2: Gráfico Gantt de la planificación inicial.

10.2. Desarrollo real del proyecto.

En este apartado se muestra la duración real de cada una de las etapas de las que se compone el proyecto. Al igual que en la planificación, se ofrece la duración en días y las horas de dedicación a cada una de las etapas. Como se puede apreciar en la Tabla 129, la fecha de finalización del proyecto se ha aumentado en una semana, por lo que la entrega de éste trabajo también se ha retrasado 7 días.

Actividad	Fecha de Inicio	Ficha de Fin	Duración(Días)	Horas dedicadas
Plan operativo	15-12-2014	22-12-2014	7	40
Análisis	22-12-2014	8-1-2015	18	44
Diseño	9-1-2015	30-1-2015	24	64
Implementación	2-2-2015	15-4-2015	73	156
Pruebas	2-2-2015	26-4-2015	53	124
Evaluación	27-4-2015	10-5-2015	14	36
Documentación	2-2-2015	7-6-2015	168	440
Revisión	8-6-2015	19-6-2015	12	80

Tabla 129: Planificación real.

En cuanto al diagrama de Gantt del proyecto, el resultado de la planificación real se puede observar en las Tabla 130.1 y 130.2.

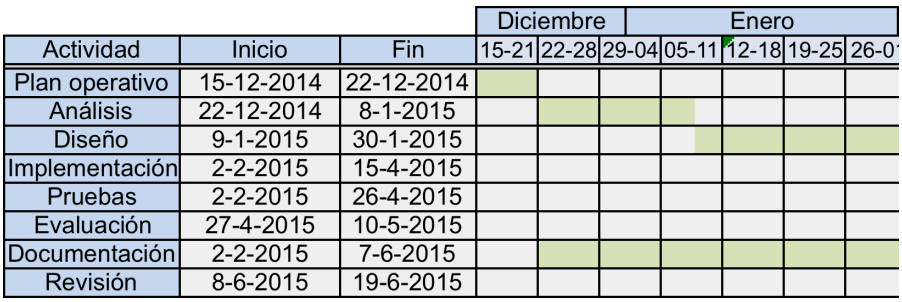


Tabla 130.1: Gráfico Gantt de la planificación real.

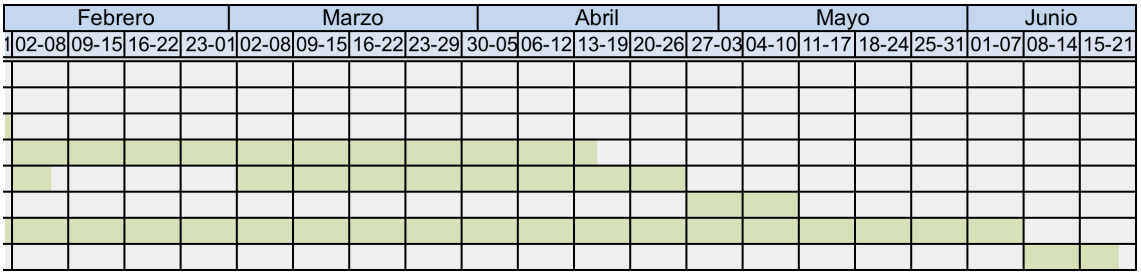


Tabla 130.2: Gráfico Gantt de la planificación real.

10.3. Presupuesto

En este apartado se presenta el presupuesto estimado del proyecto. Primero, se mostrará el gasto previsto para el proyecto. Después, se mostrará el gasto total del proyecto, y se terminará por comparar ambos gastos para ver si el gasto supera la estimación inicial.

10.3.1. Coste planificado

En este punto se desglosa el gasto previsto del proyecto, en el que se incluyen gastos en Hardware, Software y en personas.

10.3.1.1. Coste en Hardware

El gasto total en hardware asciende a 327 €, tal y como se puede comprobar en la Tabla 131. Cabe destacar el importe y la depreciación de la instancia en el Cloud, ya que su valor es de cero euros porque ha sido cedida gratuitamente por el grupo de investigación de Arquitectura de Computadores, Comunicaciones y Sistemas.

Concepto	Importe	Dedicación (Meses)	Periodo de depreciación (Meses)	Coste imputable
Ordenador MacBook pro retina 15'	2.030,00 €	7	48	296,04 €
Ratón inalábrico Magic Mouse	69,00 €	7	48	10,06 €
Adaptador de Mini DisplayPort a VGA	29,00 €	7	48	4,23 €
Monitor BenQ GL2250HM	114,30 €	7	48	16,67 €
Instancia tipo large de arcos compute cloud	0,00 €	7	0	0,00 €
Total				**327,00 €**

Tabla 131: Gastos Hardware.

10.3.1.2. Coste en Software

El gasto total en Software asciende a 39,23 €, tal y como se puede comprobar en la Tabla 132.

Concepto	Importe	Dedicacion (Meses)	Periodo de depreciación (Meses)	Coste imputable
Microsoft word 2011 para Mac	269,00 €	7,00 €	48,00 €	39,23 €
Ubuntu 14.04	0,00 €	7,00 €	48,00 €	0,00 €
Servidor de aplicaciones Apache2	0,00 €	7,00 €	48,00 €	0,00 €
framework Twitter Bootstrap	0,00 €	7,00 €	48,00 €	0,00 €
Conjunto de librerias fpdf	0,00 €	7,00 €	48,00 €	0,00 €
Herramienta wkhtmltopdf	0,00 €	7,00 €	48,00 €	0,00 €
Heramienta pdftk	0,00 €	7,00 €	48,00 €	0,00 €
SonarQube	0,00 €	7,00 €	48,00 €	0,00 €
Sonar-runner	0,00 €	7,00 €	48,00 €	0,00 €
OpenStack	0,00 €	7,00 €	48,00 €	0,00 €
Total				**39,23 €**

Tabla 132: Gastos Software.

10.3.1.3. Coste en personal

En este apartado se desarrolla el coste planificado por el personal asociado al proyecto. Los cargos asociados son los que se muestran en la Tabla 133.

Categorías	Cargo	Responsable
C1	Analista programador	Iván Plaza Alonso
C2	Responsable del proyecto	Francisco Javier García Blas
C3	Jefe de proyecto	Jesús Carretero Pérez

Tabla 133: Cargos asociados al proyecto.

La asignación planificada, en horas, de cada uno de los cargos a cada una de las fases de la planificación queda representada en la Tabla 134.

Actividad	Horas dedicadas		
	C1	C2	C3
Plan operativo	20	10	10
Análisis	44	0	0
Diseño	64	0	0
Implementación	143	0	0
Pruebas	143	0	0
Evaluación	36	10	10
Documentación	390	0	0
Revisión	0	50	14
Total horas por categorías	840	70	34
Coste hora / categoría	21,50 €	35,23 €	48,96 €
Coste por categoría	18.060,00 €	2.466,10 €	1.664,64 €
Coste Total			**22.190,74 €**

Tabla 134: Cargos asociados al proyecto.

El coste total planificado en personal asociado al proyecto es de 22.190,74 €.

10.3.1.4. Coste total planificado

Una vez presentados todos los costes, el coste total planificado del sistema asciende a 32.752,72 € (TREINTA Y DOS MIL SETECIENTOS CINCUENTA Y DOS EUROS CON SETENTA Y DOS CENTIMOS) I.V.A INCLUIDO, tal y como se puede apreciar en la Tabla 135.

Concepto	Coste
Costes Hardware	327,00 €
Costes Software	39,23 €
Costes Personal	22.190,74 €
Beneficio (20%)	4.511,39 €
Total sin I.V.A.	27.068,36 €
I.V.A.	5.684,36 €
Total con I.V.A.	**32.752,72 €**

Tabla 135: Coste total planificado.

10.3.2. Coste real del proyecto

En este punto se desglosa el gasto real que ha sufrido el proyecto. En este coste se incluyen gastos en Hardware, Software y en personas.

10.3.2.1. Coste en Hardware

El coste total en Hardware asciende a 327 €. Este gasto es igual al expuesto en la planificación inicial.

10.3.2.2. Coste en Software

El gasto total en Software asciende a 39,23 €. Este gasto coincide con el coste planificado.

10.3.2.3. Coste en personal

En este apartado se desarrolla el coste real que ha producido el personal asociado al proyecto. Los cargos asociados son los que se muestran en la Tabla 136.

Categorías	Cargo	Responsable
C1	Analista programador	Iván Plaza Alonso
C2	Responsable del proyecto	Francisco Javier García Blas
C3	Jefe de proyecto	Jesús Carretero Pérez

Tabla 136: Cargos asociados al proyecto.

La asignación planificada, en horas, de cada uno de los cargos a cada una de las fases reales queda representada en la Tabla 137.

Actividad	Horas dedicadas		
	C1	C2	C3
Plan operativo	20	10	10
Análisis	44	0	0
Diseño	64	0	0
Implementación	156	0	0
Pruebas	124	0	0
Evaluación	16	10	10
Documentación	440	0	0
Revisión	0	60	20
Total horas por categorías	864	80	40
Coste hora / categoría	21,50 €	35,23 €	48,96 €
Coste por categoría	18.576,00 €	2.818,40 €	1.958,40 €
Coste Total			**23.352,80 €**

Tabla 137: Cargos asociados al proyecto.

El coste total planificado en personal asociado al proyecto es de 23.352,80 €.

10.3.2.4. Coste real del proyecto y comparación con el coste planificado

Una vez presentados todos los costes reales, el coste total del Proyecto asciende a 32.752,72 € (TREINTA Y DOS MIL SETECIENTOS CINCUENTA Y DOS EUROS CON SETENTA Y DOS CENTIMOS) I.V.A INCLUIDO, tal y como se puede apreciar en la Tabla 138. Como se puede observar, el coste real coincide con el coste planificado, ya que al cliente se le ofrece un presupuesto inicial y debe ser respetado. Como el proyecto se ha entregado una semana después de lo planificado, para poder mantener el presupuesto inicial, se ha bajado el beneficio del proyecto, pasando de un 20,00% a un 14,12%.

Concepto	Coste
Costes Hardware	327,00 €
Costes Software	39,23 €
Costes Personal	23.352,80 €
Beneficio (14,12%)	3.349,33 €
Total sin I.V.A.	27.068,36 €
I.V.A.	5.684,36 €
Total con I.V.A.	**32.752,72 €**

Tabla 138: Coste total planificado.

10.4. Entorno Socio-económico

Para realizar el análisis del entrono socio-económico se utilizará el Análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) y se establecerá un posible plan de ventas para evaluar si el proyecto generará beneficios en los años posteriores a su creación.

10.4.1. ANÁLISIS DAFO

El análisis mediante DAFO representa una forma sencilla y eficaz para identificar el futuro de proyectos e incluso empresas. Además, ayuda a plantear acciones para aprovechar todas las oportunidades identificadas y preparar planes para mantener, en este caso, el sistema en el tiempo.

10.4.1.1. Debilidades

La principal debilidad que presenta el sistema con el resto de alternativas en el mercado (por ejemplo, SonarQube, Cppcheck PMD) es que todas estas alternativas cuentan con grandes grupos de desarrollo que constantemente están actualizando los sistemas. Además, la cuota de mercado puede estar muy condicionada al enfocar el proyecto al ámbito educativo.

10.4.1.2. Amenazas

La principal amenaza la representa SonarQube, ya que cuenta con una alta cuota de mercado. Por otra parte, las grandes empresas de desarrollo software, como Indra, cuentan con sistemas internos y comités internos de evaluación de código. Además, muchas herramientas de análisis de código cuentan con *plugins* para diferentes ambientes de desarrollo integrados (IDE), que facilitan seguir la evolución de la calidad de los productos que se desarrollan, ya que cuentan con resultados en tiempo real.

10.4.1.3. Fortalezas

El proyecto realizado representa una solución multiplataforma real, ya que al estar desarrollada como plataforma web, se permite el acceso desde cualquier dispositivo con conexión a internet. Además, el sistema ofrece al usuario una interfaz muy sencilla, la cuál siempre muestra las opciones mas destacadas, como subir proyectos, realizar análisis, visualizar informes, etc., para que el usuario tenga el control en todo momento.

10.4.1.4. Oportunidades

El sistema se enfocará hacia la pequeña y mediana empresa dedicada al software, ya que supone un gran sector en el desarrollo del software y por lo general no están sujetos a ningún tipo de comité de evaluación de sus productos software. Además su gestión de los proyectos muchas veces consiste en desarrollar soluciones en un tiempo muy corto, por lo que la calidad del software pasa a ser una opción y no una necesidad real. Finalmente la solución puede tener una gran aceptación en el sector educativo.

10.4.2. PLAN DE VENTAS

Dado que el competidor directo, SonarQube, ofrece una serie de *plugins* para extender las funcionalidades del sistema con un coste por unidad de 7.000 € anuales. En este apartado se desarrollan una serie de posibles escenarios para el sistema teniendo en cuenta que el precio por licencia es de 40 € mensuales, 480 € anuales.

Mes	Gastos	Ingresos			
		100	200	400	800
dic-14	4.542,14 €	0,00 €	0,00 €	0,00 €	0,00 €
ene-15	9.084,29 €	0,00 €	0,00 €	0,00 €	0,00 €
feb-15	13.626,43 €	0,00 €	0,00 €	0,00 €	0,00 €
mar-15	18.168,57 €	0,00 €	0,00 €	0,00 €	0,00 €
abr-15	22.710,72 €	0,00 €	0,00 €	0,00 €	0,00 €
may-15	27.252,86 €	0,00 €	0,00 €	0,00 €	0,00 €
jun-15	31.795,00 €	0,00 €	0,00 €	0,00 €	0,00 €
jul-15	31.795,00 €	400,00 €	800,00 €	1.200,00 €	1.600,00 €
ago-15	31.795,00 €	800,00 €	1.600,00 €	2.400,00 €	3.200,00 €
sept-15	31.795,00 €	1.200,00 €	2.400,00 €	3.600,00 €	4.800,00 €
oct-15	31.795,00 €	1.600,00 €	3.200,00 €	4.800,00 €	6.400,00 €
nov-15	31.795,00 €	2.000,00 €	4.000,00 €	6.000,00 €	8.000,00 €
dic-15	31.795,00 €	2.400,00 €	4.800,00 €	7.200,00 €	9.600,00 €
ene-16	31.795,00 €	2.800,00 €	5.600,00 €	8.400,00 €	11.200,00 €
feb-16	31.795,00 €	3.200,00 €	6.400,00 €	9.600,00 €	12.800,00 €
mar-16	31.795,00 €	3.600,00 €	7.200,00 €	10.800,00 €	14.400,00 €
abr-16	31.795,00 €	4.000,00 €	8.000,00 €	12.000,00 €	16.000,00 €
may-16	31.795,00 €	4.400,00 €	8.800,00 €	13.200,00 €	17.600,00 €
jun-16	31.795,00 €	4.800,00 €	9.600,00 €	14.400,00 €	19.200,00 €
jul-16	31.795,00 €	5.200,00 €	10.400,00 €	15.600,00 €	20.800,00 €
ago-16	31.795,00 €	5.600,00 €	11.200,00 €	16.800,00 €	22.400,00 €
sept-16	31.795,00 €	6.000,00 €	12.000,00 €	18.000,00 €	24.000,00 €
oct-16	31.795,00 €	6.400,00 €	12.800,00 €	19.200,00 €	25.600,00 €
nov-16	31.795,00 €	6.800,00 €	13.600,00 €	20.400,00 €	27.200,00 €
dic-16	31.795,00 €	7.200,00 €	14.400,00 €	21.600,00 €	28.800,00 €
ene-17	31.795,00 €	7.600,00 €	15.200,00 €	22.800,00 €	30.400,00 €
feb-17	31.795,00 €	8.000,00 €	16.000,00 €	24.000,00 €	32.000,00 €
mar-17	31.795,00 €	8.400,00 €	16.800,00 €	25.200,00 €	33.600,00 €
abr-17	31.795,00 €	8.800,00 €	17.600,00 €	26.400,00 €	35.200,00 €
may-17	31.795,00 €	9.200,00 €	18.400,00 €	27.600,00 €	36.800,00 €
jun-17	31.795,00 €	9.600,00 €	19.200,00 €	28.800,00 €	38.400,00 €
jul-17	31.795,00 €	10.000,00 €	20.000,00 €	30.000,00 €	40.000,00 €
ago-17	31.795,00 €	10.400,00 €	20.800,00 €	31.200,00 €	41.600,00 €
sept-17	31.795,00 €	10.800,00 €	21.600,00 €	32.400,00 €	43.200,00 €
oct-17	31.795,00 €	11.200,00 €	22.400,00 €	33.600,00 €	44.800,00 €
nov-17	31.795,00 €	11.600,00 €	23.200,00 €	34.800,00 €	46.400,00 €
dic-17	31.795,00 €	12.000,00 €	24.000,00 €	36.000,00 €	48.000,00 €

Tabla 139: Gastos e ingresos asociados al sistema durante los dos primeros años.

En la Tabla 139 se puede apreciar el gasto asociado al sistema junto con los ingresos que supondría durante los dos primeros años de vida. Para poder observar estos datos mas gráficamente, se proporciona la Ilustración N. En esta ilustración se puede comprobar como en los dos primeros años con tan solo 100 y 200 licencias no se obtendría la inversión inicial. Sin embargo, con 400 licencias se obtendría la inversión

inicial en Agosto del 2017. Para 800 licencias, se obtendría la inversión inicial en febrero de 2017.

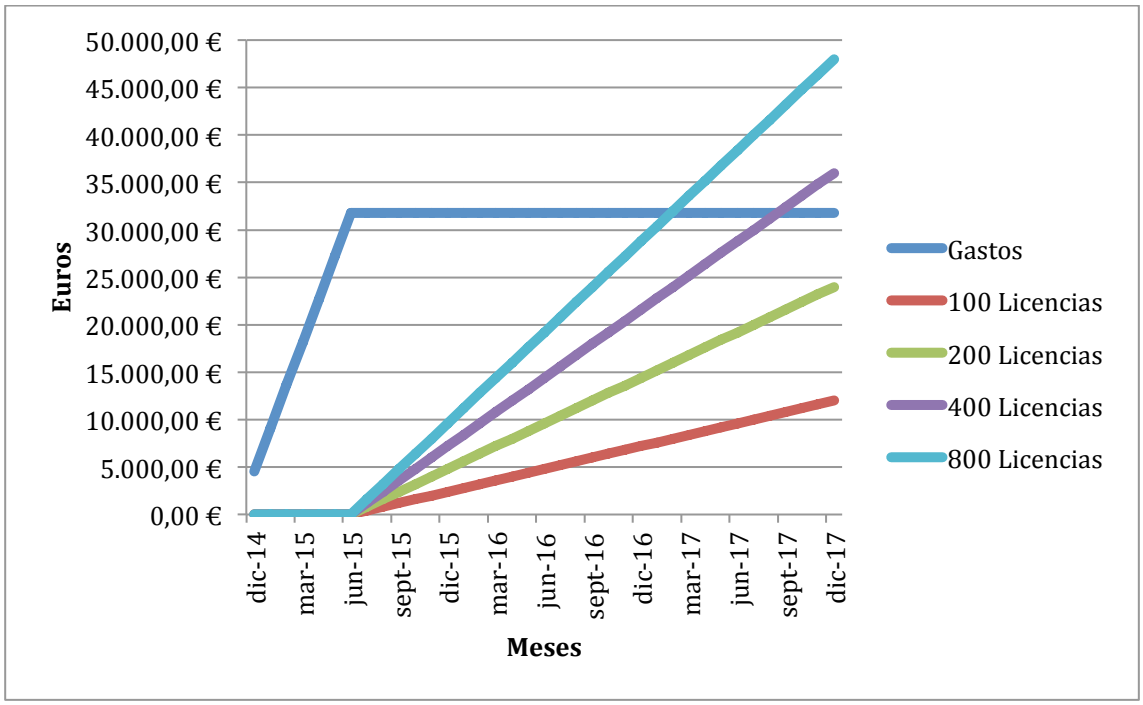


Ilustración 47: Gastos e ingresos asociados al sistema durante los dos primeros años.

CAPITULO 11
BIBLIOGRAFÍA

[1] Un enfoque actual sobre la calidad del software. *Eprints Repository Software* [en línea] [sin fecha]. [Consulta: 12 December 2015]. Disponible en: http://eprints.rclis.org/5424/1/aci05395.htm.

[2] ADÁN, V.G. and POSTED BY VÍCTOR GÓMEZ ADÁN 2012. La calidad del software. *Globe Testing* [en línea]. [Consulta: 12 December 2014]. Disponible en: http://www.globetesting.com/2012/07/la-calidad-del-software/.

[3] GARZÁS, J. 2013. Top 8 de errores informáticos más costosos de la historia - Javier Garzás. En: J. GARZÁS, *Javiergarzas* [en línea]. [Consulta: 12 December 2014]. Disponible en: http://www.javiergarzas.com/2013/05/top-7-de-errores-informaticos.html.

[4] C, M. [sin fecha]. Activities - MISRA C. *MISRA* [en línea]. [Consulta: 15 December 2014]. Disponible en: http://www.misra.org.uk/misra-c/Activities/MISRAC/tabid/160/Default.aspx.

[5] ISO 25000 PORTAL. *ISO 25000 Software product quality* [en línea] [sin fecha]. [Consulta: 15 December 2014]. Disponible en: http://iso25000.com/index.php/en/.

[6] Incibe. *Incibe* [en línea] [sin fecha]. [Consulta: 1 June 2015]. Disponible en: https://www.incibe.es/extfrontinteco/img/File/intecocert/EstudiosInformes/demostrador/monografico_cumplimiento_legal.pdf.

[7] Documento BOE-A-2007-18243. [en línea] [sin fecha]. [Consulta: 1 June 2015]. Disponible en: http://www.boe.es/buscar/doc.php?id=BOE-A-2007-18243.

[8] Documento consolidado BOE-A-1996-8930. [en línea] [sin fecha]. [Consulta: 1 June 2015]. Disponible en: http://www.boe.es/buscar/act.php?id=BOE-A-1996-8930.

[9] Software » OpenStack Open Source Cloud Computing Software. [en línea] [sin fecha]. [Consulta: 15 December 2014]. Disponible en: https://www.openstack.org/software/.

[10] PARRILLA, D. 2011. Blog Archive » Un vistazo rápido a la historia de OpenStack. En: D. PARRILLA, *Nubeblog* [en línea]. [Consulta: 12 December 2014]. Disponible en: http://nubeblog.com/2011/01/10/un-vistazo-rapido-a-la-historia-de-openstack/.

[11] Main Page. *OpenStack Wiki* [en línea] [sin fecha]. [Consulta: 15 December 2014]. Disponible en: https://wiki.openstack.org/wiki/Main_Page.

[12] OPENSTACK [sin fecha]. OpenStack Installation Guide for Ubuntu 12.04/14.04 (LTS) - icehouse. [en línea]. [Consulta: 15 December 2014]. Disponible en: http://docs.openstack.org/icehouse/install-guide/install/apt/content/ch_overview.html#architecture_overview.

[13] EUCALYPTUS 2014. eucalyptus/eucalyptus. [en línea]. [Consulta: 16 December 2014]. Disponible en: https://github.com/eucalyptus/eucalyptus/wiki.

[14] vCloud Suite, funciones de cloud privada basadas en vSphere: VMware. *VMWare* [en línea] 2014. [Consulta: 16 December 2014]. Disponible en: http://www.vmware.com/es/products/vcloud-suite.

[15] IBM Bluemix Docs. [en línea] [sin fecha]. [Consulta: 16 December 2014]. Disponible en: https://www.ng.bluemix.net/docs/#overview/overview.html.

[16] SonarQubeTM. [en línea] [sin fecha]. [Consulta: 16 December 2014]. Disponible en: http://www.sonarqube.org.

[17] Cppcheck A tool for static C/C++ code analysis. [en línea] [sin fecha]. [Consulta: 16 December 2014]. Disponible en: http://cppcheck.sourceforge.net.

[18] PMD – Welcome to PMD. [en línea] [sin fecha]. [Consulta: 16 December 2014]. Disponible en: http://pmd.sourceforge.net/pmd-5.3.2/.

[19] CodePro Analytix User Guide. [en línea] [sin fecha]. [Consulta: 16 December 2014]. Disponible en: https://developers.google.com/java-dev-tools/codepro/doc/.

[20] Introducción al HTML. *Mozilla Developer Network* [en línea] [sin fecha]. [Consulta: 17 December 2014]. Disponible en: https://developer.mozilla.org/es/docs/Web/Guide/HTML/Introduction_alhtml.

[21] DESARROLLOWEB.COM [sin fecha]. Javascript a fondo. [en línea]. [Consulta: 17 December 2012]. Disponible en: http://www.desarrolloweb.com/javascript/#quees.

[22] PHP: ¿Qué es PHP? - Manual. [en línea] [sin fecha]. [Consulta: 17 December 2014]. Disponible en: https://php.net/manual/es/intro-whatis.php.

[23] DESARROLLOWEB.COM and ÁLVAREZ, M.A. [sin fecha]. Qué es JSP. [en línea]. [Consulta: 17 December 2014]. Disponible en: http://www.desarrolloweb.com/articulos/831.php.

[24] Apache HTTP Server Project. [en línea] [sin fecha]. [Consulta: 18 December 2014]. Disponible en: http://httpd.apache.org.

[25] Beginner's Guide. [en línea] [sin fecha]. [Consulta: 17 December 2014]. Disponible en: http://nginx.org/en/docs/beginners_guide.html.

[26] Live Activity Monitoring of NGINX Plus. [en línea] [sin fecha]. [Consulta: 17 December 2014]. Disponible en: http://nginx.com/products/live-activity-monitoring/.

[27] BECERRO, A. [sin fecha]. Introducción a Shell Script. *El viajero* [en línea]. [Consulta: 17 December 2014]. Disponible en: http://www.elviajero.org/antoniux/tutos/shell_intro.pdf.

[28] gnuplot homepage. [en línea] [sin fecha]. [Consulta: 17 December 2014]. Disponible en: http://www.gnuplot.info.

[29] Visualization: Column Chart. *Google Developers* [en línea] [sin fecha]. [Consulta: 17 December 2014]. Disponible en: https://google-developers.appspot.com/chart/interactive/docs/gallery/columnchart.

[30] PÉREZ, A. [sin fecha]. Estructuración y Especificación de Casos de Uso - alfonsoperezr. [en línea]. [Consulta: 22 December 2014]. Disponible en: https://sites.google.com/site/alfonsoperezr/investigacion/estructuracin-y-especificacin-de-casos-de-uos.

[31] GAUDIN, O. [sin fecha]. Requirements - SonarQube. [en línea]. [Consulta: 29 December 2014]. Disponible en: http://docs.sonarqube.org/display/SONAR/Requirements.

[32] Entendiendo Modelo-Vista-Controlador. *CakePHP* [en línea] [sin fecha]. [Consulta: 9 January 2015]. Disponible en: http://book.cakephp.org/1.3/es/The-Manual/Beginning-With-CakePHP/Understanding-Model-View-Controller.html.

[33] En: E. BAHIT, *POO yMVC en PHP* [en línea] [sin fecha]. [Consulta: 12 January 2015]. Disponible en: http://collection.openlibra.com.s3.amazonaws.com/pdf/eugeniabahitpooymvcenphp.pdf?AWSAccessKeyId=AKIAIGY5Y2YOT7GYM5UQ&Signature=SWCLDvQdQmGDLznDQKVEb18YOPM%3D&Expires=1433846843.

[34] Qué es un Diagrama de Flujo - Gestión de Procesos Aiteco Consultores. [en línea] [sin fecha]. [Consulta: 13 January 2015]. Disponible en: http://www.aiteco.com/que-es-un-diagrama-de-flujo/.

[35] Diagrama de componentes. *ARQHYS Arquitectura* [en línea] [sin fecha]. [Consulta: 16 January 2015]. Disponible en: http://www.arqhys.com/general/diagrama-de-componentes.html.

[36] GAUDIN, O. [sin fecha]. Installing - SonarQube. [en línea]. [Consulta: 2 February 2015]. Disponible en: http://docs.sonarqube.org/display/SONAR/Installing.

[37] RACODON, D. [sin fecha]. Running SonarQube as a Service on Linux - SonarQube. [en línea]. [Consulta: 2 February 2015]. Disponible en: http://docs.sonarqube.org/display/SONAR/Running+SonarQube+as+a+Service+on+Linux.

[38] RACODON, D. [sin fecha]. Installing and Configuring SonarQube Runner - SonarQube. [en línea]. [Consulta: 2 February 2015]. Disponible en: http://docs.sonarqube.org/display/SONAR/Installing+and+Configuring+SonarQube+Runner.

[39] SONARSOURCE [sin fecha]. SonarSource/sonar-examples. [en línea]. [Consulta: 2 February 2015]. Disponible en: https://github.com/SonarSource/sonar-examples/blob/master/scripts/database/mysql/create_database.sql.

[40] MANDRIKOV, O.E.- [sin fecha]. Analyzing with SonarQube Runner - SonarQube. [en línea]. [Consulta: 8 June 2015]. Disponible en: http://docs.sonarqube.org/display/SONAR/Analyzing+with+SonarQube+Runner.

[41] Activar SSL en Apache2 (Ubuntu 10.04). *BetaTwits* [en línea] 2010. [Consulta: 8 April 2015]. Disponible en: https://betatwits.wordpress.com/2010/07/21/activar-ssl-en-apache2-ubuntu-10-04/.

ANEXO 1
ENGLISH COMPETITIONS

This annex contains the English competitions, required for finishing the Bachelor´s degree in Computer Science and Engineering at Carlos III University.

First, we will introduce the project objectives, the principal motivation, and the structure that the project follows. Then, we will show the system results with were extracted from three ARCOS assignments. Finally, we will present the project conclusions and the future lines to preserve the system.

1. INTRODUCTION

This chapter contains a formal presentation of the document. This chapter describes, in a general form, the principal motivation that has promoted the project creation, as well as, the objectives that it targets and the project structure.

1.1. MOTIVATION

Nowadays, most of the software companies invest a lot of its budget to analyse the software quality. There are two types of software analyses: static and dynamic (these concepts are defined at point 2.2. Code analyser, this point belongs to state-of-the-art). The principal propose of quality software measurement is to generate full operative applications without unusual errors.

The software quality is defined as a group of quality metrics that defines the software utility. In other words, the software quality defines its efficiency, flexibility, portability, and security in terms of the code analysis. The quality can be measured and sometimes depends on the system, the application target, and the final utility. For this reason, software that is used on aviation will be use a quality profile, in where no error are allowed. Others applications uses a lower restrictive quality profile.

The software quality measure could be done after the system or application generation. However, it can be generate a high cost if some issues are detected, and these issues implies to modify the initial design. For this reason, it is recommended a parallel quality methodology during the software generation, helping to developers to not spread issues during de software life cycling.

Along the history, the expenses that have been produced by software issues have been incremented gradually. An example of expensive software issue is the Ariane 5's rocket first flight in 1996. The rocket changed its route 40 seconds after the launch, then the rocket was divided and exploded. The committee that investigated the accident said that an issue in the inertial reference system software produced the error. This issue was in a specific part of the code. this code did a data conversion; a 64-bit float value was converted to 16-bit integer value. Furthermore, the software checks never were doing and the issue never was detected.

This kind of error still trendy right now. Other famous example was in 2010: Toyota was obligated to do an internal investigation above their brake system. Later, when the investigation ended, 270,000 automobiles were inspected because the brake software was an error in the calibration subsystem.

To deal with this type of errors, there are a lot of software quality standards, all of them was proposed to measure and to correct quality issues. An example of quality standard is MISRA. It guidelines was generated by Motor Industry Software Reliability

Association. Other example is the ISO/IEC 25000 standards, which is an international standard that evaluates the software quality and quality characteristics.

For these reasons, the main motivation for this Final Degree Project is the creation of a platform that evaluates source code and allows following and controlling the evolution of projects by static code analysis.

1.2. Objectives

The main objective for this Final Degree Project is the creation of a platform that allows to developers to evaluate their software in order to control the projects evolution.

To accomplish this objective, it is necessary to complete the following secondary objectives:

- Users management and organization. The user data are private and can't be viewed by anyone.
- Provide a subsystem, which manages the project and versions, uploads.
- Make easy the code analysis upload to users.
- Make easy the software quality following to users.
- Deploy the system in a flexible platform.

1.3. Structure

This document contains 10 chapters. The chapter sequence follows the order established by Carlos III University of Madrid for Finals Degree Projects. Additionally, the document follows the software development general structure. The principal purpose of this structure is to facilitate the document reading and to allow following the project development. The document structure as follows:

- Chapter 1 Introduction: this chapter explains the motivations that have started the Project development. Also, it exposes the needed objectives to successfully accomplish the project realization.
- Chapter 2 State of the Art: this chapter presents all the tools that the project will use. Furthermore, it offers the real tools alternatives and explains the final decision of usage of each one.
- Chapter 3: System Analysis: to do the System analysis, it will do the uses cases, functional requirements, non-functional requirements and the traceability matrix which shows that all uses cases are covered by al least one requirement.
- Chapter 4: System Design: this chapter will do the detailed system design. First, it shows the high-level architecture that the cloud system follows and the project high-level architecture. Then, it will show the flowcharts that the system

follows. Next, it will do the component diagrams. Finally, it will do the sequence diagrams that it is directly derived to the component diagrams.

- Chapter 5 Implementation: this chapter explains the process followed to implement the System: it starts from the creation of the virtual instance and ends with the PHP classes' codification.
- Chapter 6: SonarQube rules expansions: this chapter explains how to extend the rules that SonarQube has.
- Chapter 7 Testing: this phase specifies the test plan that the system will pass. Besides, it will show the traceability matrix to verify that all requirements are covered by at least one test.
- Chapter 8 Tool evaluation: the tool or system will be evaluated by three laboratories analyses, all laboratories belong to subjects of the ARCOS research group, the principal aim is to get results that indicate if the system works well and the system performance.
- Chapter 9 Conclusions and future lines: this chapter shows all the conclusions that have been deduced during the project realization. Also, these conclusions will be compares with the initial objectives, which were marked at the beginning and it shows the future lines that will be ensure that permanence of the system in time.
- Chapter 10: Planning and Budget: It will show the initial planning and it will compare with the real system development. To accomplish that, each stage planed schedule will be compare with the real used time to do it. Finally, this chapter presents the planning and the real budget of the system.
- Chapter 11 References: it contains all the references that are used during the development of this project.

2. TOOL EVALUATION

This chapter includes the evaluation carried out. The evaluation has helped by the Computer Architecture and Technology Area group, ARCOS, of Carlos III University of Madrid. The evaluation covers the following laboratories, which were implemented by students:

- Operating Systems, 2nd course, 3rd assignment , group 89.
- Operating System Design, 3rd course, 1st assignment , group 89.
- Distributed Systems, 3rd course, 2nd assignment number 2, group 89.

First, each assignment will be analyzed independently, in order to get the relation between the duration of the analysis and the code complexity and the code size. Then, the results and the mean values of each assignment will be compared in order to get an idea about the relation about the code quality and the analysis time.

2.1. ASSIGNMENT NUMBER 3 OF OPERATING SYSTEMS

This assignment is composed of 19 submissions. The associated values to the time, complexity, and the code line size are presented on the Table 140. The analysis time mean, by submission, for this assignment is 11.32 seconds. And the total time, which is inverted to analyse the assignment, is 215.089 seconds.

Are very important the values that were obtained in the practises 1, 11 and 15. Each practise only have time value and don't have values in another metrics. These null values are produced by an error while SonarQube is analysing the code, all of these practises have some compiling errors, for this reason SonarQube fails and the code quality analysis never ends.

Project	Time	Complexity	Lines
1	6,1871872		
2	12,5411839	81	836
3	12,7832699	77	721
4	12,1447070	65	697
5	11,6294241	33	448
6	13,0692029	43	607
7	12,1684799	42	533
8	12,6077611	37	423
9	11,6678050	58	530
10	11,8211319	49	562
11	6,2809699		
12	12,4530361	70	771
13	12,9262121	54	496
14	12,5555248	65	626
15	5,9860792		
16	12,4171839	47	584
17	11,5592320	37	423
18	12,0064900	56	565
19	12,2851191	81	628

Table 140: Results obtained in the assignment number 3 of Operating Systems.

2.1.1. OBTAINED DATA INTERPRETATION

In this section, the data, which were obtained previously by the assignment analysis, will be interpreted. To do this, all the obtained time value will be compared with the other values (complexity, and lines of code).

As you can see in the Figure 48, the analysis total duration of each practise is independent of the code complexity. Also, the time neither depends on the size of each practise, as it doesn't have an evident relation in both values.

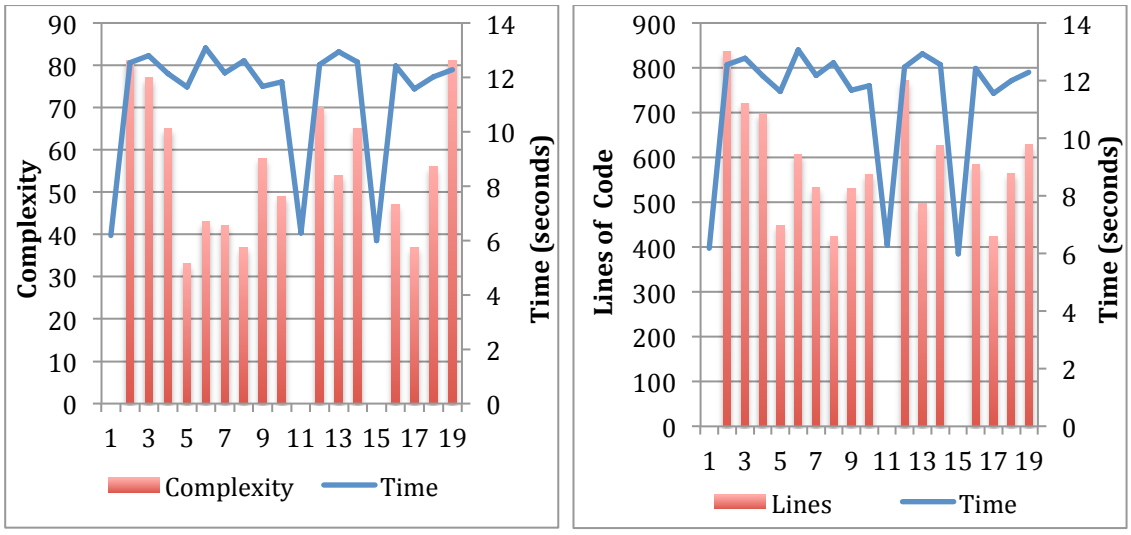


Figure 48: Graphic representation of time with the rest of values.

2.2. Assignment number 2 of Distributed Systems

In this section the code given by the Distributed System Curse students, group 89, is analysed. This assignment has got 12 groups of practises. The analysis result of this assignment is shown in Table 142.

Project	Time	Complexity	Lines
1	19,2969220	628	4200
2	12,1466749	53	422
3	6,1637690		
4	11,8107688	70	726
5	12,7816169	67	723
6	11,5033100	62	642
7	14,1306419	84	570
8	3,9272180		
9	6,3603978		
10	12,4792118	98	714
11	12,2198899	55	497
12	13,5447462	51	540

Table 141: Results obtained in the assignment number 2 of Distributed Systems.

2.2.1. Obtained data interpretation

As the previous assignment, there are some practises that have compiling errors, so, the SonarQube analysis fails. As you can see in Figure 49, there isn't an evident relation with the time and the complexity or the time and the size of each practise.

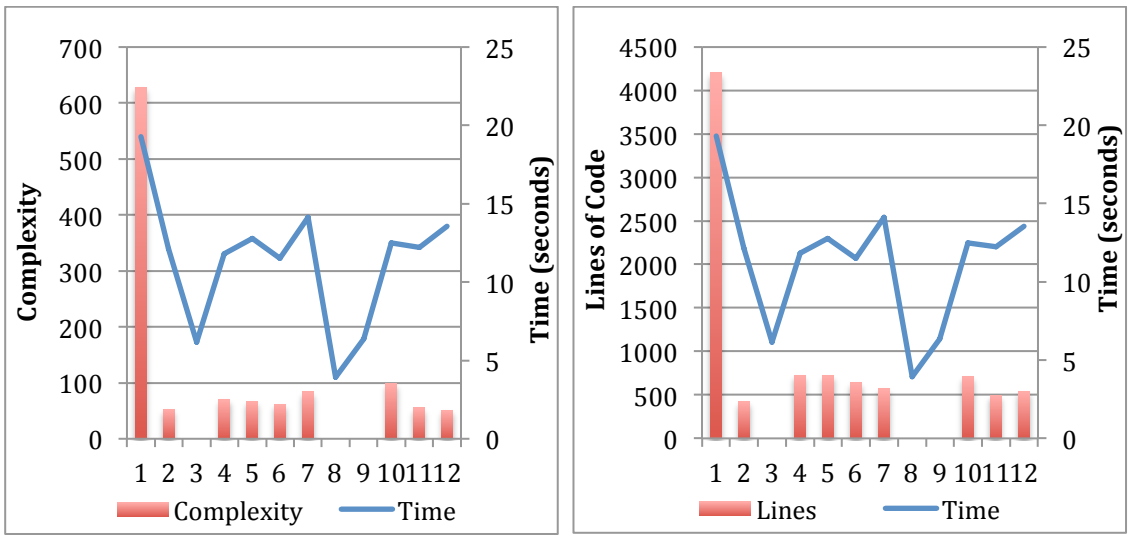


Figure 49: Graphic representation of time with the rest of values.

2.3. Assignment number 1 of Operating Systems Design

The Operating System Design laboratory has 11 practises. In this case, only one of these practises has compiling errors. Besides, the second practise has very high values at time, complexity and errors, as you can see in Table 143.

Project	Time	Complexity	Lines
1	14,7422478	166	1044
2	36,5421009	4751	24554
3	14,6628120	160	1092
4	14,2120390	161	933
5	14,1339760	179	1020
6	12,9928651	155	1005
7	13,3709970	184	1006
8	12,7067931	110	587
9	5,9983931		
10	13,2825010	161	933
11	13,6749928	110	587

Table 143: Results obtained in the assignment number 1 of Operating System Design.

2.3.1. Obtained data interpretation

This laboratory has different results as the rest laboratories. The relation between time and the code size is most evident. However, the duration is independent of the code complexity, as you can see in Figure 50.

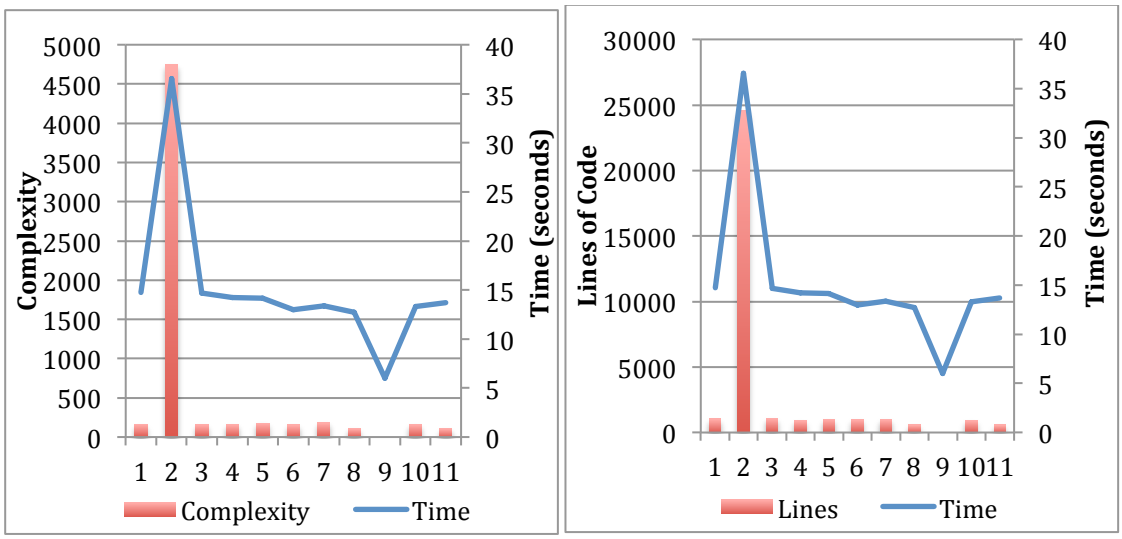


Figure 50: Graphic representation of time with the rest of values.

2.4. OVERVIEW OF DATA RESULTS

This section shows an analysis of the means and deviations, which are calculated for all the laboratories.

As you can see in Table 128, the mean time for a small code (about 1000 lines of code) is 11,3 seconds. Its high deviation is determined by the practices that don't compile, which have a low analysing time.

The complexity mean value grow when the system analyses assignments of later courses. It is produced because every year the practises are more difficult than the other years and it requests more effort for students, for these reasons the mean value and the standard deviation will increases with the courses.

Finally, the lines of code mean value and its standard deviation indicate that data are so dispersed. It indicates that for any assignment, with a concrete functionality, the size of the code changes so much between practises.

Assignment	Time		Complexity		Lines of Code	
	Mean	Deviation	Mean	Deviation	Mean	Deviartion
SSOO Lab1	11,3205	2,338811	55,9375	15,922599	590,625	120,54923
SSDD Lab2	11,363764	4,1290858	129,77778	187,46051	1003,7778	1203,3926
DSSOO Lab1	15,119974	7,5042002	613,7	1453,9159	3276,1	7478,4548

Table 144: Mean and standard deviation for the data of the three laboratories.

3. CONCLUSIONS AND FUTURE LINES

This chapter will cover all the conclusions that have extracted from the created system. Also, in this chapter, we discusse about all the future lines that would improve the system.

3.1. CONCLUSIONS

At the beginning of this document, the following objectives were marked:

- Users management and organization. The user data are private and can't be viewed by anyone.
- Provide a subsystem, which manages the project and versions, uploads.
- Make easy the code analysis upload to users.
- Make easy the software quality following to users.

Deploy the system in a flexible platform.After the system development has finished, we can say that all the objectives, which were exposing at the beginning of this document, have been completed satisfactorily. Furthermore, this project has allowed applying the knowledge, which was acquired along the study at University.

Moreover, the use of cloud computing, currently rising, has been a great asset, because it is a technology that the Bachelor´s degree in Computer Science and Engineering don't cover in its subjects. Also, the PHP programming language, which is widely used at the business world, has verifying the learning curve, which is needed to write operational code with a new language.

Finally, the system will need some improves in the future to keep correctly the system. Besides, each component will need some updates to include a lot of features and additional security.

3.2. FUTURE LINES

This section shows and discusses the futures lines that will improve the system presented in this Final Bachelor´s degree project.

3.2.1. PARALLELIZE DATABASES ACCESSES

By default, SonarQube with MySQL as a database manager and InnoDB as a storage mechanism lacks of simultaneous data access. Therefore, when the system attempts to perform a parallel analysis, it produces an error and the analyses fails. The SonarQube component is open source and it allows to modify the code to introduce locks in the data access sections.

3.2.2. Include nodes to perform code analysis in Windows

The developed system permits analysing projects, which are writing in C#, under a small rules profile. Also, SonarQube allows adding rules from FxCop repository. FxCop is a tool that analyses .NET code but its principal disadvantage is that it targets Windows operating system. For these reason, a possible improvement would be to create a pool of machines with SonarQube systems running under different operating systems in order to provide a precise analysis for each programming language. Besides, it is able to install a large number of tools to increase the profiles and the rules.

3.2.3. Creating a custom profiles

The SonarQube application allows including new static analyse rules. One possible improvement would be to implement a profile with the good practice rules like MISRA (for C and C++), ISO/IEC 25000 or custom profiles for students to verify their practices code quality.

ANEXO 2
INTERFAZ DE USUARIO

En este anexo se presenta la interfaz de usuario generada para el sistema. La primera pantalla del sistema es la de acceso, como se puede apreciar en la Ilustración 51.

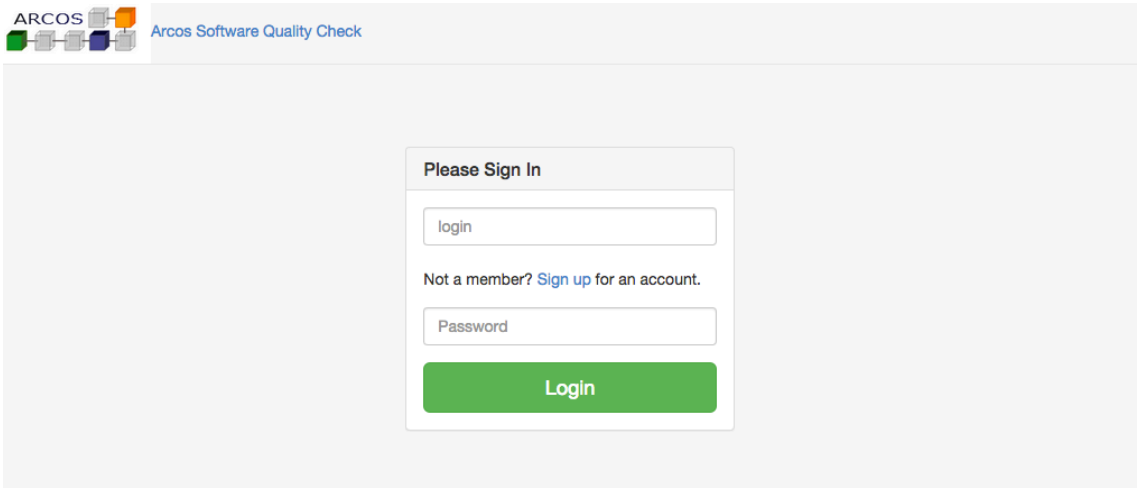


Ilustración 51: Acceso al sistema.

Para que los usuarios accedan al sistema, cuentan con una sección de registro desde la página de acceso. Los datos que deben insertar se muestra en la Ilustración 52.

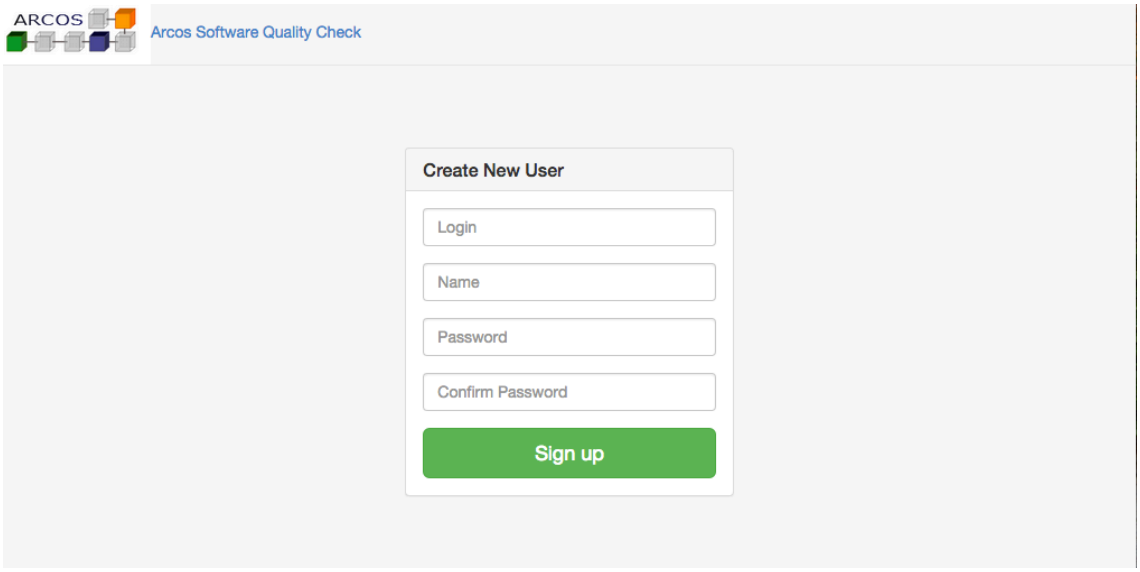


Ilustración 52: Registro de usuarios

Una vez registrado en el sistema se concede el acceso, la página principal del sistema muestra todas las funciones que se pueden realizar dentro del sistema. Además, se muestran dos marcadores que indican el número total de proyectos subidos al sistema y el número de proyectos analizados.

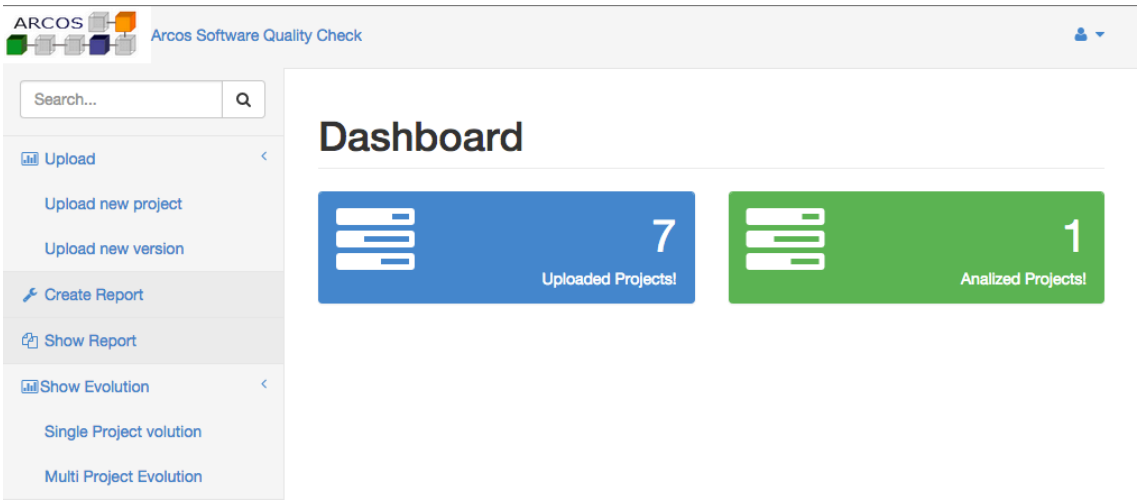


Ilustración 53: Página inicial del sistema.

Una de las ventanas destacables es la evolución de proyectos, en la cuál se puede observar el gráfico de barras que marca la evolución de los errores y la tabla situada debajo de ésta, la cuál muestra los distintos valores de la complejidad asociada al proyecto.

Ilustración 54: Pantalla de evolución de un proyecto.

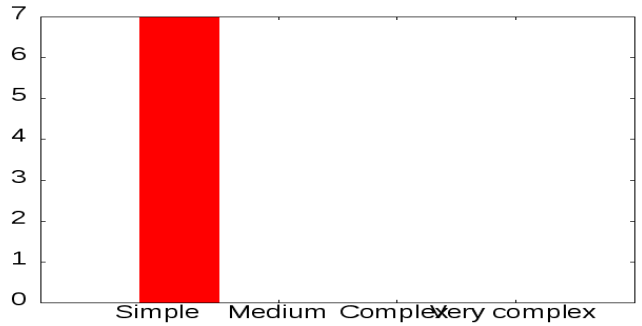
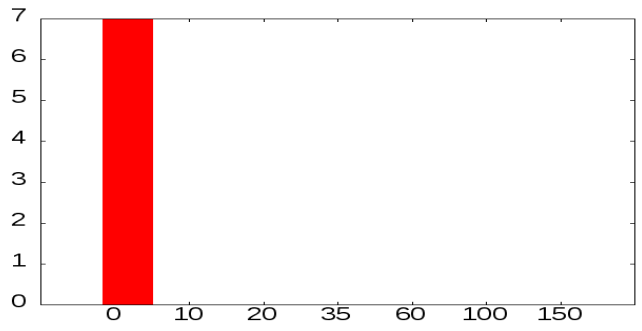
ANEXO 3
INFORME GENERADO

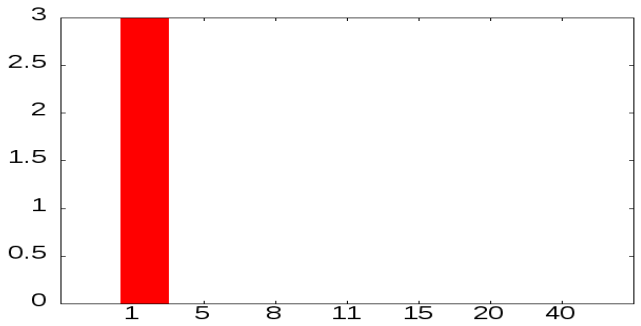
En este anexo se muestra un ejemplo de informe generado por el sistema. El informe se compone de tres partes:

- Summary: resumen de los valores más destacables de las métricas.
- Issues Report: líneas de código que incumplen alguna de las reglas de calidad del *software*.
- Profile Rules: perfil de reglas por el que ha pasado el código analizado.

Summary

ac_p3_2013-2014 - version: 1.0

Name	Value
Abacus complexity	Simple
Abacus complexity distribution	
Blocker issues	0
Number of comment lines	0
Comments balanced by ncloc + comment lines	0
Cyclomatic complexity	16
Confirmed issues	0
Critical issues	0
development_cost	2430
Directories	1
Duplicated blocks	0
Duplicated files	0
Duplicated lines	0
Duplicated lines balanced by statements	0
False positive issues	0
Complexity average by file	2.3
Files distribution /complexity	
Number of files	7
Complexity average by function	2.3

Functions distribution /complexity	
Functions	7
Info issues	0
Lines	108
Non Commenting Lines of Code	81
Non Commenting Lines of Code Distributed By Language	c=81
Open issues	32
Reopened issues	0
sqale_rating	1
Number of statements	51
Issues	32

Issues Report

ac_p3_2013-2014 - 18-jun-2015 9:59:47

New issues	Resolved issues	Issues
32	0	32

Issues per Rule	New issues	Resolved issues	Issues
Variable 'varname' is assigned a value that is never used	11	0	11
Variable 'varname' is not assigned a value	7	0	7
Insufficient comment density	7	0	7
Uninitialized variable: varname (since Cppcheck 1.52)	6	0	6
The scope of the variable 'varname' can be reduced	1	0	1

Only NEW issues | Filter by:

ac_p3_2013-2014/test1-1.c

4 New issues	0 Resolved issues	4 Issues

Variable 'varname' is not assigned a value	1	0	1
Uninitialized variable: varname (since Cppcheck 1.52)	1	0	1
Variable 'varname' is assigned a value that is never used	1	0	1
Insufficient comment density	1	0	1

3 more comment lines need to be written to reach the minimum threshold of 25.0% comment density. | NEW

Insufficient comment density

```
2   {
3    int a = 42;
4    int b[1000000];
```

Variable 'b' is not assigned a value. | NEW

Variable 'varname' is not assigned a value

```
5    int i;
6
7    for(i=0; i < 1000000; i++){
8     a = a + b[0] + 1;
```

Variable 'a' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

Uninitialized variable: b | NEW

Uninitialized variable: varname (since Cppcheck 1.52)

```
9    }
10   }
```

ac_p3_2013-2014/test1-2.c

4 New issues	0 Resolved issues	4 Issues

Variable 'varname' is not assigned a value	1	0	1
Uninitialized variable: varname (since Cppcheck 1.52)	1	0	1
Variable 'varname' is assigned a value that is never used	1	0	1
Insufficient comment density	1	0	1

3 more comment lines need to be written to reach the minimum threshold of 25.0% comment density. | NEW

Insufficient comment density

```
2   {
3    int a = 42;
4    int b[1000000];
```

Variable 'b' is not assigned a value. | NEW

Variable 'varname' is not assigned a value

```
5    int i;
6
```

```
7   for(i=0; i < 1000000; i++){
8    a = a + b[i] + 1;
```

Uninitialized variable: b | NEW

Uninitialized variable: varname (since Cppcheck 1.52)

Variable 'a' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
9   }
10  }
```

ac_p3_2013-2014/test1-3.c

	New issues	Resolved issues	Issues
	5	0	5
Variable 'varname' is not assigned a value	1	0	1
Uninitialized variable: varname (since Cppcheck 1.52)	1	0	1
Variable 'varname' is assigned a value that is never used	1	0	1
The scope of the variable 'varname' can be reduced	1	0	1
Insufficient comment density	1	0	1

4 more comment lines need to be written to reach the minimum threshold of 25.0% comment density. | NEW

Insufficient comment density

```
2   {
3    int a = 42;
4    int b[1000000];
```

Variable 'b' is not assigned a value. | NEW

Variable 'varname' is not assigned a value

```
5    int random_index = 3;
```

The scope of the variable 'random_index' can be reduced. | NEW

The scope of the variable 'varname' can be reduced

```
6    int i;
7
10   for(i=0; i < 1000000; i++){
11    random_index = rand() % 1000000;
12    a = a + b[random_index] + 1;
```

Variable 'a' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

Uninitialized variable: b | NEW

Uninitialized variable: varname (since Cppcheck 1.52)

```
13  }
14  }
```

ac_p3_2013-2014/test1-4.c

	New issues	Resolved issues	Issues
	4	0	4
Variable 'varname' is not assigned a value	1	0	1
Uninitialized variable: varname (since Cppcheck 1.52)	1	0	1
Variable 'varname' is assigned a value that is never used	1	0	1
Insufficient comment density	1	0	1

3 more comment lines need to be written to reach the minimum threshold of 25.0% comment density. | NEW

Insufficient comment density

```
2   {
3    int a = 42;
4    int b[1000000];
```

Variable 'b' is not assigned a value. | NEW

Variable 'varname' is not assigned a value

```
5   int i;
6
7   for(i=0; i < 1000000; i++){
8    a = a + b[i%16] + 1;
```

Uninitialized variable: b | NEW

Uninitialized variable: varname (since Cppcheck 1.52)

Variable 'a' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
9   }
10  }
```

ac_p3_2013-2014/test2.c

	New issues	Resolved issues	Issues
Variable 'varname' is assigned a value that is never used	5	0	5
Variable 'varname' is not assigned a value	1	0	1
Uninitialized variable: varname (since Cppcheck 1.52)	1	0	1
Insufficient comment density	1	0	1

6 more comment lines need to be written to reach the minimum threshold of 25.0% comment density. | NEW

Insufficient comment density

```
2   {
3   int a = 42;
4   int b[1048576];
```

Variable 'b' is not assigned a value. | NEW

Variable 'varname' is not assigned a value

```
5   int i;
6
11
12  for(i=0; i < 512*8; i++){
13   a = a + b[0] + 1;
```

Uninitialized variable: b | NEW

Uninitialized variable: varname (since Cppcheck 1.52)

Variable 'a' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
14   c = c + b[32768] + 1;
```

Variable 'c' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
15   d = d + b[32768*2] + 1;
```

Variable 'd' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
16   e = e + b[32768*3] + 1;
```

Variable 'e' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
17   f = f + b[32768*4] + 1;
```

Variable 'f' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
18  }
19  }
```

ac_p3_2013-2014/test3.c

	New issues	Resolved issues	Issues
Variable 'varname' is not assigned a value	1	0	1
Uninitialized variable: varname (since Cppcheck 1.52)	1	0	1

Variable 'varname' is assigned a value that is never used	1	0	1
Insufficient comment density	1	0	1

4 more comment lines need to be written to reach the minimum threshold of 25.0% comment density. | NEW

Insufficient comment density

```
3   {
4    int a = 42;
5    int b[1000000];
```

Variable 'b' is not assigned a value. | NEW

Variable 'varname' is not assigned a value

```
6    int i;
7    int param = atoi(argv[1]);
9
10   for(i=0; i < 1000000; i++){
11    a = a + b[i%param] + 1;
```

Uninitialized variable: b | NEW

Uninitialized variable: varname (since Cppcheck 1.52)

Variable 'a' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
12   }
13  }
```

ac_p3_2013-2014/test4.c

3 New issues	0 Resolved issues	3 Issues
Variable 'varname' is not assigned a value	1	0	1
Variable 'varname' is assigned a value that is never used	1	0	1
Insufficient comment density	1	0	1

5 more comment lines need to be written to reach the minimum threshold of 25.0% comment density. | NEW

Insufficient comment density

```
4   {
5    int a = 42;
6    int b[1048576];
```

Variable 'b' is not assigned a value. | NEW

Variable 'varname' is not assigned a value

```
7    int i;
8    int z;
12    for(z = 0; z < 16; z++){
13     for(x = 0; x < 1024; x++){
14                  a = a + b[i*16384 + x*16 + z];
```

Variable 'a' is assigned a value that is never used. | NEW

Variable 'varname' is assigned a value that is never used

```
15              }
16    }
```

Priority	Name
Major	The time handling functions of library time.h shall not be used (MISRA 20.12)
Major	An unconditional break statement shall terminate every non-empty case clause of a switch (MISRA 15.2)
Major	All uses of the #pragma directive shall be documented and explained (MISRA 3.4)
Major	#undef shall not be used (MISRA 19.6)
Critical	The right hand operand of a logical && or ll operator shall not contain side effects (MISRA 12.4)
Major	Avoid function with too many lines of code
Major	Continue statement must not be used (MISRA 14.5)
Major	Avoid file with too many lines of code
Major	Avoid function with too many parameters
Major	Sections of code should not be "commented out" (MISRA 2.4)
Major	Avoid too complex function
Major	Switch statements must use braces (MISRA 14.8)
Major	Identifiers in an inner scope shall not hide outer scope ones by reusing the same name (MISRA 5.2)
Major	Switch statements without any "case" shall be refactored (MISRA 15.5)
Major	Numeric variables being used within a for loop for iteration counting shall not be modified in the body of the loop (MISRA 13.6)
Blocker	Functions shall be declared at file scope (MISRA 8.6)
Major	Trigraphs shall not be used (MISRA 4.2)
Major	The comma operator shall not be used (MISRA 12.10)
Major	The operands of a logical && or ll shall be primary-expressions (MISRA 12.5)
Major	Assignment operators shall not be used in sub-expressions (part of MISRA 12.2)
Major	Avoid switch statement without a "default" clause (MISRA 15.3)
Major	Avoid use of string functions which do not check for buffer overflows
Major	The signal handling facilities of signal.h shall not be used (MISRA 20.8)
Major	Goto statement must not be used (MISRA 14.4)
Major	The input/output library stdio.h shall not be used in production code (MISRA 20.9)
Major	Avoid use of non-reentrant POSIX functions in favor of their reentrant version
Major	If-else statements must use braces (MISRA 14.9)
Major	While and Do/While loops must use braces (MISRA 14.8)